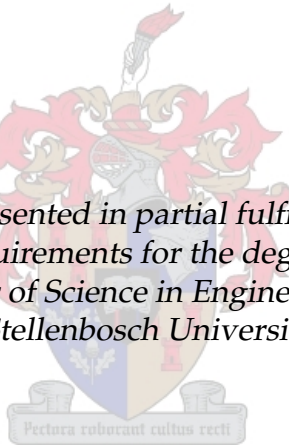
Volumetric Data Throughput Optimisation by Dynamic FEC bearing Frame Length Adaptation

by

Christian Christelis



Thesis presented in partial fulfilment of
the requirements for the degree of
Master of Science in Engineering
at Stellenbosch University

Supervisor: Dr. Riaan Wolhuter

Department Electrical and Electronic Engineering

March 2010

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2010

Abstract

The telecommunications link between a LEO satellite and a rural ground station with a non-tracking antenna, has a strongly varying link quality and a short communications window. The satellite acts as a store-and-forward node between ground stations. The TC-SDLP and an FTP protocol form a shallow protocol stack, which excludes unneeded protocol functionality and the resulting overhead. Coding gain, introduced by BCH FEC in the TC-SDLP, allows for link quality improvement.

The core of this thesis is an improvement of the TC-SDLP to maximise effective payload data throughput, or goodput. This improvement was achieved by creating an optimal segment length selection metric based on the BER. Since the BER is not determinable from within the TC-SDLP, the metric was twice determined; once based on the FER and finally based on time delays.

The work includes an extensive background study, which consists of space standardisation, orbital physics, error detection and correction, space data-link protocols, data throughput and culminating in the protocol stack design. The project specific link budget calculation is presented. The optimal segment length policy was mathematically determined.

A simulation model of the TC-SDLP was used as a proof of concept for the effective throughput and give a performance benchmark. Finally a TC-SDLP implementation offers a real world performance demonstration.

Uittreksel

Die telekommunikasie skakel tussen 'n lae aardomwenteling (LEO) sateliet en 'n plattelandse grondstasie met 'n nie-volg antenna, het 'n skakelkwaliteit wat in 'n groot mate varieer en 'n kort kommunikasievenster. Die sateliet tree op as 'n stoor- en- aanstuur node tussen grondstasies. Die TC-SDLP en 'n leêr oordrag protokol (FTP) vorm 'n vlak protokol stapel, wat onnodige protokol funksionaliteit en die gevolglike opkoste uitsluit. Kode aanwins, wat deur die BCH FEC in die TC-SDLP, aangebring word, verbeter die skakelkwaliteit.

Die kern van hierdie tesis is 'n verbetering van die TC-SDLP om sodoende die ware deurvoer van nuttige vragdata te maksimimeer. Hierdie verbetering is bereik deur die skep van 'n optimale segmentlengte-seleksie metode gebaseer of die bit fout tempo (BER).

Aangesien die BER nie bepaal kan word vanuit die TC-SDLP nie, is die maatstaf twee keer bepaal; die eerste keer is die bepaling gebaseer op die raamwerk fout tempo (FER) en die finale bepaling op tyd vertragings.

Die tesis sluit 'n omvattende agtergrondstudie in, wat bestaan uit ruimte standardisering, wentelbaan fisika, die opspoor en regstel van foute, ruimte inligtingskakel protokol en deurstuur van data wat uitloop op die protokol ontwerp. Daar word aangedui hoe die berekening van die begroting vir die skakel van toepassing op die spesifieke projek, gedoen is. 'n Wiskundige analise van die optimale segmentlengte s ook gedoen.

'n Simulasie model van die TC-SDLP is gebruik as 'n bewys van die konsep vir die ware deurset en gee 'n prestasie maatstaf. Laastens bied die TC-SDLP implementering 'n ware wereld prestasie demonstrasie.

Acknowledgements

I would like to express my sincere gratitude to:

- God
- Lorita Christelis - for your love and support
- Dr. Riaan Wolhuter - study leader extraordinaire
- Desiree Christelis, Marike Brits and Anita van der Spuy - proofreading

"I returned and saw under the sun that- The race is not to the swift, Nor the battle to the strong, Nor bread to the wise, Nor riches to men of understanding, Nor favor to men of skill; But time and chance happen to them all. " *Ecc 9:11*

Contents

List of Figures

List of Tables

Abbreviations

ACK acknowledgement

ASK Amplitude Shift Keying

AOS-SDLP Advanced Orbiting System - SDLP

AOS-TS Advanced Orbiting System - TS

BCH code by Bose, Chaudhuri and Hocquenghem

BER Bit Error Rate

BPSK Binary PSK

CCSDS Consultative Committee for Space Data Systems

CFDP CCSDS File Delivery Protocol

CLCW Communications Link Control Word

CLTU Communications Link Transmission Unit

COP-1 Communications Operation Procedure

CRC Cyclic Redundancy Check

DSP Digital Signal Processing

ECC Error Check Codes

ECSS European Cooperation for Space Standardization

ESA European Space Agency

FARM-1 Frame Acceptance and Reporting Mechanism

FEC Forward Error Correction

FER Frame Error Rate

FÊR FER estimation

FIFO First In First Out Queue

FOP-1 Frame Operation Procedure

FPGA Field-Programmable Gate Array

FSK Frequency Shift Keying

FTP File Transfer Protocol

GEO GEostationary Orbit

GS Ground Station

HEO High Earth Orbit

LCM Lowest Common Multiple

LDPC Low-Density Party-Check

LEO Low Earth Orbit

MCSP Multi Channel Satellite Payload

MEO Medium Earth Orbit

NACK negative acknowledgement

NASA National Aeronautics and Space Administration

OSI Open System Interconnection (Reference Model)

POSIX Portable Operating System Interface

PSK Phase Shift Keying

QNX Quick uNiX

QPSK Quadrature PSK

RISC Reduced Instruction Set Computer

RTOS Real Time Operating System

RS Reed-Solomon

SCCS Synchronisation and Channel Coding Sublayer

SH-4 SuperH-4 (CPU)

SNR Signal to Noise Ratio

SS Segmentation Sublayer

TC-SCCS TeleCommand - SCCS

TC-SDLP TeleCommand - SDLP

TC-SS TeleCommand - SS

TC-TS TeleCommand - TS

TM-SCCS TeleMetry - SCCS

TM-SDLP TeleMetry - SDLP

TM-TS TeleMetry - TS

TS Transfer Sublayer

UNIX UNiplexed Information and Computing System

Chapter 1

Introduction

The work as documented in this thesis, originated from the Multi Channel Satellite Payload (MCSP) project, as requested by the Department of Communications. Earlier work undertaken by the DSP - Telecommunications group, of the Department of Electrical and Electronic Engineering, highlighted the throughput limitations of a single narrow band ground-satellite communications link. The MCSP project sought to address this problem, by utilising simultaneous comms over multiple narrow band links. During this development, it also became clear that there could be merit in investigating throughput over individual links. This research was subsequently conducted and presented in this document. This chapter begins with a background on the University of Stellenbosch and Digital Signal Processing (DSP) research group's involvement with satellites. After that, the MCSP project is outlined. The objectives of the MCSP project, as well as this project, are discussed. Next, the research problem statement, to be investigated in this thesis is presented. This chapter is concluded by the a summary of the research method and a report outline.

1.1 Satellite Engineering at Stellenbosch University

In satellite design, a great variety of engineering knowledge is combined; especially in the fields of embedded systems, control systems and telecommunications. This makes satellite engineering a very lucrative field of study. Within Africa, Stellenbosch University has one of the richest histories of satellite engineering. After successfully developing Sunsat a few years ago, the DSP research group, in conjunction with SunSpace, developed the communications payload of the now current Sumbandilasat.

1.1.1 MCSP Project Outline

Low Earth Orbit (LEO) satellites, due to their short orbital periods, offer ground stations very short communication opportunities. Furthermore, the varying distance, between the LEO satellite and ground station, causes a varying link quality. These two factors limit the amount of data which can be transmitted during a pass. The MCSP is to offer increased bandwidth by using multiple channels, each at its own frequency, simultaneously. This will allow for large data files, such as X-ray images, to be conveyed to their destination using a LEO satellite.

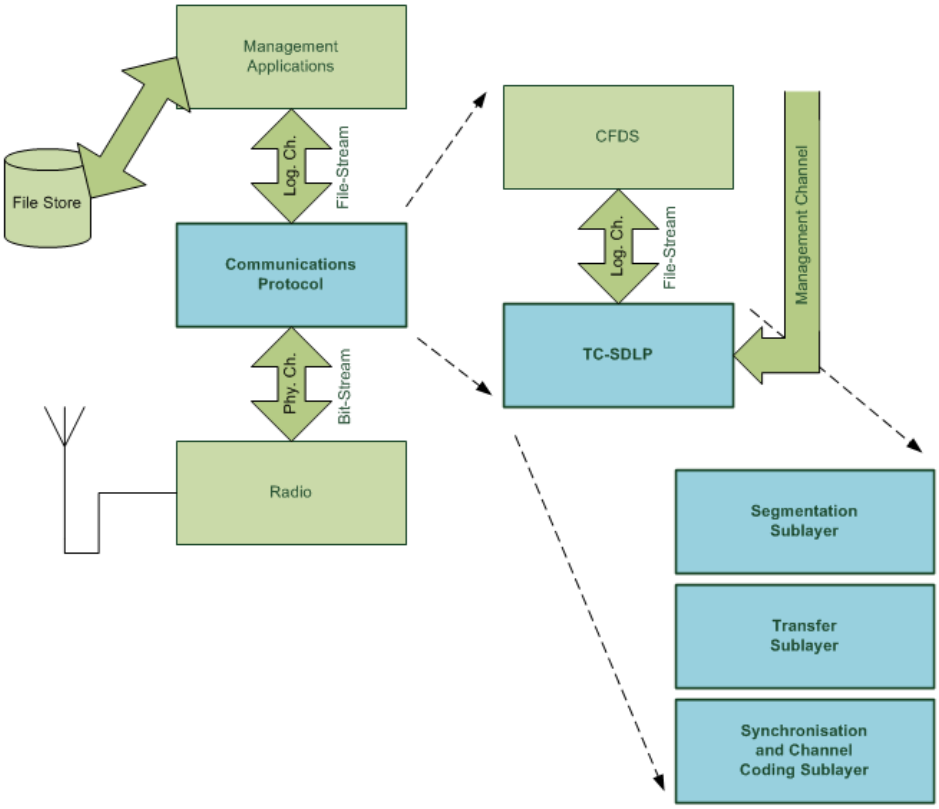


Figure 1.1.1: System Overview

A major design objective of the MCSP, as is the case with most LEO satellites, was to maximise data throughput. Redundancy was minimised by stripping down the communications requirements to the minimum. This minimum is the LEO satellite acting as a bent pipe between ground stations.

The DSP group was involved with various aspects of the communications protocol design of the MCSP project, including the design and implementation of a radio, communications protocol and the management application. These aspects of the project are shown in Figure 1.1.1. The focus of this thesis is centred around the the link layer protocol design, optimisation and implementation.

1.1.2 MCSP Objectives

The MCSP project's objectives, as per the DSP laboratory's involvement, pertain to the successful design and implementation of a prototype of the MCSP. This MCSP prototype should contain all communications and signal processing aspects of the communications channels of the MCSP project. The aspects which the prototype should cover are shown in Figure 1.1.1.

1.1.3 Project Objectives

During this project a choice regarding the communications protocol stack was made. The choice of the communications protocol stack, a CFDP running on a TeleCommand - SDLP (TC-SDLP), will be discussed in Section 2.7.1.2. The link quality is obtained by investigating influencing aspects (Section 2.4 and 3.2).

The design and implementation of the TC-SDLP, to a prototype level, as per Chapter 4, is an objective within the scope of the greater MCSP project. The central objective of this thesis is to answer the research problem as set out in Sections 3.3, 3.4 and 3.5.

1.1.4 Problem Statement

The satellite network involves communication between the satellite and a ground station only, hence the network consists of a single hop. This makes many of the layers of functionality in typical network stacks redundant. The TC-SDLP protocol renders the service of reliable and in sequence data transfer over the space link.

The TC-SDLP divides payload data into segments of variable length. Typically, a static segmentation length policy, based on a data throughput tradeoff, is used throughout the mission or a mission phase. This is a good policy, provided the link quality remains relatively constant throughout the mission or mission phase.

In the case of the MCSP, as with any LEO satellite, the link quality varies greatly during each pass and even between passes. This means that the optimal segment length to maximise effective data throughput, is not constant.

The optimal segment length for any specific link quality can be determined. The segment length, or frame length, should be adapted to the optimal segment length for the current link quality. The core problem is thus:

Maximising volumetric payload data throughput for a variable quality LEO satellite - ground station link, by optimising frame lengths of the Forward Error Correction (FEC) bearing TC-SDLP, by utilising on the fly link quality estimation.

1.1.5 Research Method

The first step taken to solve the research problem is to find the underlying model describing the optimal segment length in terms of a given link quality. This can be done mathematically by calculating the effective data throughput, after applying FEC, based on the link Bit Error Rate (BER).

Since the BER is not known within the TC-SDLP, its effects are to be measured. The metric which best maps back to the BER, is the Frame Error Rate (FER). Hence the link quality is to be described in terms of the FER. An algorithm will be developed to determine the optimal segment length from the FER.

The TC-SDLP is divided into sub-layers; each of these sub-layers has a specific task to perform and has access to a subset of system data. The FER, which describes the link quality, is found within information located in the TeleCommand - TS (TC-TS). The TeleCommand - SS (TC-SS) makes decisions about segment length. Using the FER metric located in the TC-TS, to make a decision about segmentation length in the TC-SS, requires information sharing, which may place the integrity of the TC-SDLP sub-layered architecture in jeopardy.

Thus an alternate method of acquiring the FER, based on time delays within the TC-SS, will be investigated. Hence decisions made within the TC-SS can be based exclusively on information available within this sublayer.

The proof of concept is offered by a simulation model. The simulation model makes use of the exact FER, thus proving if the FER is known, then the optimal segment length can be obtained by the optimal segment length algorithm.

The final step in researching the optimal segment length policy is modifying the TC-SDLP implementation to use the optimal segment length policy. In the implementation the estimation of frame errors, rather than the actual FER, is used. The results of the simulation as well as the overall system approach are presented, from which conclusions will be drawn.

1.2 Thesis Outline

The structure of this thesis will be as follows:

- Chapter 2 - Background Study: In Chapter 2 the literature reviewed and the theory used to create the project context, is summarised. The

protocol choice is made and motivated, therefore obtaining the design aspect of the project objective.

- Chapter 3 - System Design: In Chapter 3 the research problem statement is contextualised. The optimal segment length is calculated for a given BER by using effective data throughput calculations. An optimal segment length policy contour is found. This is extended to a model containing only data available to the Segmentation Sublayer (SS). Finally, this chapter presents a simulator which will be used to verify the real implementation. A large part of the research problem statement is investigated and most of the research methods discussed in Section 1.1.5 are applied.

- Chapter 4 - Implementation: In Chapter 4 the implementation, a substantial coding effort, is discussed. Design choices, where interesting, are highlighted.

- Chapter 5 - Results: In Chapter 5 the results which were obtained from tests on the complete prototype are presented. These are compared to the expected results obtained by calculation and from the simulation.

- Chapter 6 - Conclusion, Recommendations and Future Work: In Chapter 6 provides a summary of work done, concluded results, contributions and recommendations for future work.

Chapter 2

Background Study

Chapter 1 has given an introduction to the MCSP project as well as this project as subset of the MCSP project. This chapter presents the history and activities conducted in the fields of wireless telecommunications communications, information theory, coding, satellite dynamics, protocol engineering as well as general satellite telecommunications. This chapter creates the scaffolding used in the system design shown in Chapter 3.

The concepts used in later chapters are built up from basic components, some of which are mentioned only in passing. The focal topics discussed in this chapter include: Factors influencing Bit Error Rate, Forward Error Correction and Space Datalink Protocols.

2.1 Satellite History and Background

A satellite may be broadly defined as a celestial body orbiting another larger celestial body. The larger body, such as a planet orbited by a moon, is called the primary. The term natural satellites contrasts the term artificial satellite, where artificial satellite refers to an object placed into space by man. The term satellite will hence forth exclusively refer to artificial satellites orbiting earth.

The first artificial satellite launched was Sputnik 1. Sputnik 1 was launched by the Soviet Union on 4 October 1957. With a mass of 83.6 kg and at an altitude of 939 km at the apogee and 215 km at the perigee the satellite orbited the earth every 96.2 minutes for 3 months. Sputnik 1 communicated by emitting radio signals at 20.005 and 40.002 MHz.

Sputnik 1 ushered in a new technological era. This era was dubbed "the space age". The space race between the Soviet Union and the United States was a hallmark of the space age.

Since the launch of Sputnik 1 approximately 6000 satellites have been launched [2]. According to the NASA Orbital Debris Program Office there are more than 13000 officially catalogued objects (larger than 10cm) currently

(April 2009) orbiting the earth of which about 3200 are spacecrafts [3].

2.2 Space and Space Standards Organisations

The space industry is privileged to have many key contributing committees and organisations. Many space protocols and much space research have been published by these. Some of these prominent organisations are listed in the following sections, along with some of their individual contributions.

2.2.1 ESA

Figure 2.2.1: ESA Logo

The European Space Agency (ESA) was established in 1975 by its 11 founding member countries. It has a current membership of 18 countries. The activities of the ESA range from observing the earth and telecommunication to human space-flight. [4]

The ESA has launched numerous space crafts from its launch site in French Guiana. Kourou, the launch site, lies $5^{o}3'$ north of the equator.

2.2.2 NASA

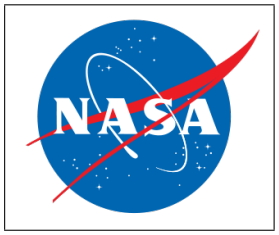


Figure 2.2.2: NASA Logo

The National Aeronautics and Space Administration (NASA) was established in 1958 by the United States of America. The activities of NASA are similar to those of the ESA, with the addition of military applications. [5]

NASA was created to join the space race. NASA's first satellite was Explorer 1. Over the years NASA has remained a great driving force in the space industry.

The NASA Orbital Debris Program tracks man-made and other debris orbiting the earth. NASA Safety Standard 1740.14 has become a mandatory part of the NASA missions assesing debris.

2.2.3 ECSS

Figure 2.2.3: ECSS Logo

The European Cooperation for Space Standardization (ECSS) was founded in 1993. The need for a uniform set of coherent space standards in Europe was the fundamental motivation for the inception of the ECSS. The discussion that follows gravitates around the documents published by the ECSS.

The document [6] is the ECSS glossary of terms. In the document a number of terms with definitions as applicable to all ECSS publications. The same definitions apply throughout this thesis.

The document [1] is the principal top-level ECSS document. An overview as well as an introduction to all other ECSS standards are given in this document. The distinction between standards, handbooks and technical memoranda is defined. The document also elaborates on the identification system used throughout the documents. The letters "P" or "S" are used for system -, "M" is used for management -, "E" is used for engineering and finally "Q" is used for quality assurance related documents. For the detailed breakdown of the numbering system reference is made to Figure 2.2.4.

Refering to Figure 2.2.4 it may be noted that the documents that are of highest concern, in regard to this project, are the "E-50" set of documents. Auxiliary document sets are the "M-10", "E-10" and "E-40".

In the following paragraphs some of the auxiliary documents are discussed. The discussion of the datalink layer specifications of the "E-70" set of documents is deferred to Section 2.6.3.

2.2.3.1 ECSS-E-10

In the engineering management document [7] the terms functional and technical specifications are defined. The standard describes methods of defining

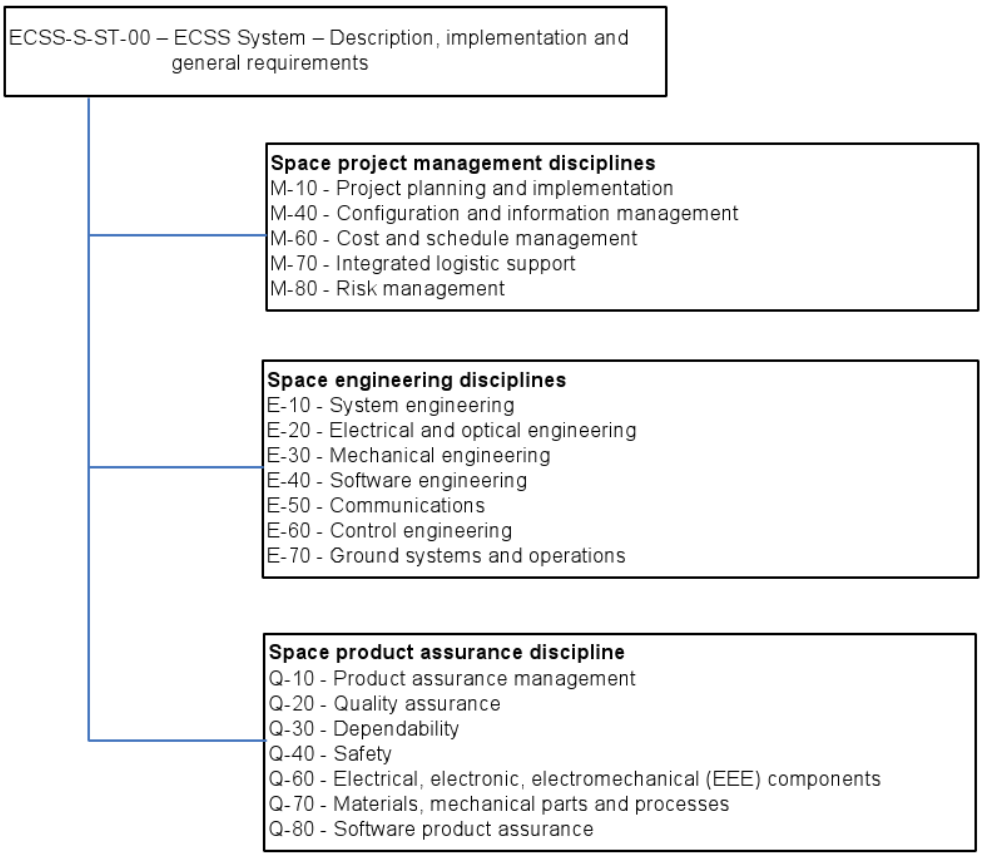


Figure 2.2.4: ECSS Standards Overview (taken from [1])

the requirements of a space project in terms of functional requirements. Furthermore, the standard investigates manners in which functional specifications are formulated into technical specifications. These functional specifications are passed to the management of the project to oversee the realisation of the functional specifications.

2.2.3.2 ECSS-M-10

The project management document [8] focuses on project management aspects of a space project. The focus of this document spans project planning, - organization, - breakdown and - phasing.

2.2.3.3 ECSS-E-40

The software engineering document [9] describes various aspects of software. The document covers the following items:

1. Software related system requirements process
2. Software management process
3. Software requirements and architecture engineering
4. Software design and implementation engineering
5. Software validation process
6. Software delivery and acceptance process
7. Software verification process
8. Software operation process
9. Software maintenance process

These aspects of software engineering encompass the entire software life cycle. The parts of the software life cycle that are important to this project are Item 4 and 5.

The software design and implementation process encompasses the three steps of: Designing the software items, coding and testing and the integration of the items. The validation process assures that the requirements were implemented correctly and without omission.

2.2.4 CCSDS

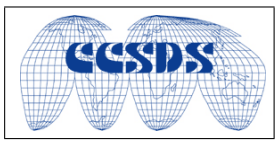


Figure 2.2.5: CCSDS Logo

The Consultative Committee for Space Data Systems (CCSDS) was founded in 1982. The CCSDS was founded to facilitate international discussion on space communications issues. Many of the standards created by this organisation were incorporated into the ECSS standard base. CCSDS standards have been used extensively in space missions, more than 400 to date [10].

The CCSDS uses a colour coded system to identify its document. The colours indicate the following type of report:

- Blue: Recommended Standards
- Magenta: Recommended Practices

- Green: Informational Reports
- Orange: Experimental
- Yellow: Record
- Silver: Historical

The documents most used in this work were all green books pertaining to space communications protocols. A survey of these documents gives the reader a sound overview of the specific standard. This report typically includes the summary of concepts as well as the rationale behind a standard.

The blue books contain actual standards. Many of the standards presented in these books have equivalent documents in the ECSS document library. In some cases the ECSS library refers to CCSDS documents. In the case of a such a reference the CCSDS blue books were used. These cases, where applicable to this work, have been noted by the reference used.

2.3 Satellite Dynamics

Investigating and understanding the movement and dynamics of a satellite it is a logical first step in satellite engineering. This section looks at the physical forces acting upon the satellite.

2.3.1 Orbit

The orbit of a satellite was first described by the German mathematician Johannes Kepler (1571 - 1630). Kepler developed three laws in the field of celestial mechanics describing the motion of planets around the sun. Kepler's work in celestial mechanics can easily be applied to satellites when replacing the words sun with earth and planet with satellite.

The orbit of a satellite, with the earth at one of the foci or the focus according to Kepler's first law, is described by the orbit equation:

$r(\theta) = \frac{h^2}{\mu} \frac{1}{1+e\cos\theta}$ where

- r is the distance between the centre of the earth and the satellite
- h is the relative angular momentum
- e is the eccentricity of the orbit
- $\mu = GM_{earth} = 3.9866 * 10^{14}$
- θ is the angle at the focus between the current satellite position and the perigee

The orbital shape my be described in terms of its eccentricity. If the eccentricity of a satellite is $e = 0$ then the orbit is circular. If the eccentricity is $0 < e < 1$ the orbit is elliptical. In the case that the eccentricity is $e = 1$ or $1 < e$ the orbit is parabolic and hyperbolic respectively. Hence for satellites the eccentricity is bounded by $0 \leq e < 1$. It must be noted, that since the earth and the satellite are not point masses, but have radii, that r must always evaluate to greater than the sum of these two radii. Another point worth noting, is the fact that the finer points affecting satellite motion such as drag caused by atmospheric gasses and the gravitational pull of other celestial bodies are assumed negligible.

2.3.2 Classification of Orbits

The orbits that satellites move in are classified in terms of altitude and synchronicity. Altitude classifications define an orbit space that around the earth that is a given altitude above mean sea level. This does not restrict a satellite to such an altitude. Based on the eccentricity of the orbit of a satellite it can regularly move between altitude orbits. Synchronous orbits are based on how the satellite appearances at a given location coincide with the time on earth at that location.

2.3.2.1 Altitude

The altitude of the satellite affects both the velocity and the orbit period. The Section 2.3.3 describes the affects on the satellites' velocity. Orbits have been classified by altitude into 4 distinct categories. The categories are listed below with altitudes given above mean sea level.

- LEO - the LEO ranges from the an altitude of 0 km to 2000 km
- Medium Earth Orbit (MEO) - the MEO ranges from an altitude of to 2000 km to just below 35,786 km.
- GEostationary Orbit (GEO) - the GEO has an altitude of 35,786 km.
- High Earth Orbit (HEO) - the HEO has an altitude of above 35,786 km

2.3.2.2 Synchronicity

The synchronicity of a satellite is something that may be a lucrative aspect in choosing the characteristics of a satellites orbit. Three important synchronous orbits have been listed below:

- GEO - rotational period of 23 hours, 56 minutes, 4.091 seconds
- Semi-synchronous Orbit - rotational period of just under 12 h

- Sun-synchronous Orbit - an orbit which causes the satellite to passes over a given point on the earth at the same local solar time

2.3.3 Velocity

Kepler's third law states: "The square of the orbit period of a planet is proportional to the cube of its semi-major axis of its orbit." The third law specifically describes the heliocentric orbit of planets, but the principle can be applied to the Geocentric orbit of satellites. As an example, take satellite A orbiting earth X times further than satellite B; by the third law, satellite A will have an orbit period $\sqrt{X^3}$ longer than than satellite B. If following the same orbital shape the distance travelled by satellite A is X times greater than that of satellite B. Expressed as a fraction, the increase in orbiting time over the increase in distance, the fraction is $\frac{\sqrt{X3}}{X}$ or simplified, the ratio is $\sqrt{X}$. From the preceding it is clear, that the further satellites are from earth the slower they travel. The same arguments can be made using the much younger "Universal law of Gravitation".

2.3.3.1 Escape Velocity

By Newton's "Universal law of Gravitation" the gravitational potential force acting on the satellite, by the earth, can be calculated as $F_g = \frac{Gm_1m_2}{r}$. If the gravitational force is to be overcome, it is necessary to have a greater force that is normal to the gravitational force. The normal force can be equated to the gravitational force $ma = \frac{Gm_1m_2}{r}$. By replacing $a = \frac{dv}{dt} = \frac{dv}{dr}\frac{dr}{dt} = v\frac{dv}{dr}$ then integrating and simplifying $v = \sqrt{\frac{2GM}{r}}$ is obtained. The velocity described is known as the escape velocity.

2.3.3.2 Altitude and Velocity

In Section 2.3.2.1 it was mentioned that the altitude of the satellite affects the velocity at which the velocity moves. The velocity that satellites travel at in the LEO is approximately 8 $\frac{km}{s}$ where at GEO it is only about 3 km/s. The time it takes for a LEO satellite to revolve around the earth is approximately 1.5 h. This is contrasted to satellites at GEO, in which the satellite orbits the earth in 23 h 56 min 4.091 s. GEO thus match the average rotation period of the earth and make the satellites appear to maintain a constant position relative to the surface of the earth.

2.4 Radio Frequency Considerations

This section recaps wireless telecommunication. The factors affecting the strength of an electromagnetic signal, used to transmit a message, are mentioned.

2.4.1 Signal Loss

A signal attenuates as it propagates through space and matter. Attenuation is caused by a variety of reasons. In the case of a LEO satellite at operating in the S band with direct line of sight the most prominent cause of attenuation may be ascribed to the following:

- The multipath and fresneal loss may be ignored.
- The free space path loss is a constant factor, to be considered as a function of distance as well as frequency.
- Gaseous attenuation, cloud and fog attenuation, as well as rain attenuation are very much weather dependant.

$$FSPL = (\frac{4\pi df}{c})^2 \tag{2.4.1}$$

The free space path is mathematically expressed by Equation 2.4.1, by the inverse square law [11]. Weather dependant attenuation is a stochastic process, which has not explicitly been quantified in this research. The reader is made aware of its presence and detail may be found in the literature.

2.4.2 Noise

Noise is introduced into the system, when signals other than the intended signal are received. Since the Ground Station (GS) may be in a remote location, with its antennas pointing skyward, and the satellite is in space the chance of man made signals interference is relatively small. The most prominent noise source of the GS is introduced by that part of the cosmic background noise, which is allowed to pass through the band pass filter.

2.4.3 Signal to Noise Ratio

Both the signal strength and the noise strength give an indication of the perceived signal quality at the receiver. To best describe the perceived received signal quality, the Signal to Noise Ratio (SNR) is used. The SNR can be calculated by comparing the attenuated signal to the amount of received noise.

2.4.4 Modulation Scheme

Analog noise, as discussed in the preceding sections, is added to a modulated signal. When the signal is demodulated a decision is made about the intended original digital data, this decision may be correct or incorrect depending on the SNR as well as the modulation scheme used. Some common modulation schemes, along with their probability of error, are discussed in the following text. Additional detail may be found in [11].

2.4.4.1 ASK

Amplitude Shift Keying (ASK) is a modulation technique whereby binary data is modulated on the amplitude of a signal. An incorrect decision is made when the amplitude of the incoming signal is on the incorrect side of the threshold due to added noise. The development of the probability of error (P_E) expression is omitted. The expression is given in Equation 2.4.2, where the variable is energy per bit to noise power spectral density ratio ($\frac{E_b}{N_0}$).

$$P_E = Q(\sqrt{\frac{E_b}{N_0}}) \tag{2.4.2}$$

The Q function is given by Equation 2.4.3 below:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt \tag{2.4.3}$$

2.4.4.2 FSK

Frequency Shift Keying (FSK) is a modulation technique whereby binary data is modulated on the frequency of the carrier signal. The probability of error is given by Equation 2.4.4.

$$P_E = Q(\sqrt{\frac{E_b}{N_0}}) \tag{2.4.4}$$

2.4.4.3 PSK

Phase Shift Keying (PSK) is a modulation technique whereby binary data is modulated on the phase of the carrier signal. The phase of the carrier signal is shifted 180^o in the case of Binary PSK (BPSK) and 90^o for Quadrature PSK (QPSK) and $\frac{360^o}{2^N}$ for 2^N-PSK.

For BPSK Equation 2.4.5 has been developed for the probability of error. The QPSK probability of error is given by Equation 2.4.6.

$$P_E = Q(\sqrt{2\frac{E_b}{N_0}}) \tag{2.4.5}$$

$$P_E = 2Q(\sqrt{\frac{E_s}{N_0}}) \tag{2.4.6}$$

Comparing these two equations on face value would make it appear as though QPSK fares 3 dB worse than BPSK but such a straight forward comparison does not take into account the additional data contained in each symbol. The performance of QPSK and BPSK are in actual fact equal (see Chapter 8 of [11]).

2.4.5 Bit Error Rate

The probability of any bit in a message signal to be corrupted because of noise introduced in the physical layer is known as the BER. This is an important concept, because it describes the quality of the digital link. The digital link is the bit stream which is created by the dalatlink layer. The BER depends on the demodulation scheme and the $\frac{E_b}{N_0}$. To convert the analog SNR to the digital $\frac{E_b}{N_0}$, the band width of the signal as well as the baud rate is required. The greater the signal band width, the greater the $\frac{E_b}{N_0}$, on the other hand the greater the baud rate the smaller $\frac{E_b}{N_0}$. The link budget calculation, as relevant to this work, is presented in Section 3.2.

2.5 Error Correction and Detection

Due to the BER, inevitable introduction of errors in a message signal effort is often made at detecting and correcting said errors. This section presents methods that attempt to detect and correct errors in message signals.

In the applied mathematics branch, information theory, various methods have been developed to assure correctness of the information transfered between two nodes. This section investigates various FEC codes and Error Check Codes (ECC).

2.5.1 Hamming Distance

The principal behind FEC and ECC is to add redundant data to information bits producing a code word. These code words have a certain Hamming distance between each other. Hamming distance indicates the amount of errors required to be introduced into a code word or block to change it into another code word or block. If code words have a Hamming distance of d between them, then at least $d-1$ errors can be detected and up to $\frac{d-1}{2}$ errors can be corrected. The simplest example in which a code word is produced from a data block, is by utilising the well known even or odd parity check. Parity checks add a Hamming distance of 1 between code words. The trade-off to consider when choosing the required redundancy is how much redundancy and additional computation is worth the increase in reliability.

2.5.2 Coding Gain

In the literature a commonly used concept is that of coding gain. Using FEC on data, gives the decoder the possibility to correct a certain number of errors that may have been introduced during transmission. A corrected code word improves the perceived BER. The same improvement in BER could be achieved by increasing the signal strength of the transmitting node. The increased signal strength would improve the SNR, i.e. decreasing in BER.

Hence there are two methods of improving the BER: Increasing the gain of the signal and adding FEC. The coding gain offered by a FEC is equal to the signal gain, which would achieve the same improvement in the received signal.

2.5.3 Shannon Limit

It stands to reason that as the BER increases, the capacity of the channel decreases. The communications capacity compared to channel quality was determined by Claud Shannon in 1948. The Shannon limit or Shannon capacity describes the upper bound or the maximum capacity of a communications channel. The formula defining this limit may be found in the literature. [12]

2.5.4 FEC Techniques

In the field of information theory, FEC techniques have been the subject of much study for several years. The increase in computational power as well as the increased number of gates, as well as the drop in cost of Field-Programmable Gate Array (FPGA)s has brought many of these techniques out of theory into practice.

The investigation of FEC techniques in this section begins with block codes, specifically code by Bose, Chaudhuri and Hocquenghem (BCH), Reed-Solomon (RS) and Low-Density Party-Check (LDPC) codes. Next a FEC technique, which encodes a message of arbitrary length, Convolutional code, will is investigated. Finally two combination coding techniques Concatenated - and Turbo codes are discussed.

2.5.4.1 BCH

Discovered by Hocquenghem in 1959 and independently by Bose and Chaudhuri in 1960, BCH codes are named after the inventors. BCH codes are systematic block codes, based on Finite, or Galois, Field calculations. Extensive coverage provided by Lin and Costello may be found in [12]. The information presented in this section originates from this very complete book.

2.5.4.1.1 Galois Fields were defined by Évariste Galois (1811 - 1832). A Galois Field is denoted by $GF(N)$ where $N = p^k$ and p is prime. GF(N) contains all integers up to N-1. A Galois Field is a special group, over which the operations of addition, subtraction, multiplication and division can be done without leaving the field.

Two important operations, defined over Galois Fields, are modulo-N addition and modulo-N multiplication. It has been shown that both of these operations are complete over the Galois Fields [12]. Modulo-N operations are cyclic operations, since for any $a \in GFN$ $a + N = a$. The results of modulo-2

addition and multiplication operations are shown in Table 2.5.1 and 2.5.2. As with normal addition and multiplication there is an inverse operation to the modulo-2 operations.

Table 2.5.1: Modulo-2 Addition

$\oplus$	0	1
0	0	1
1	1	0

Table 2.5.2: Modulo-2 Multiplication

$\odot$	0	1
0	0	0
1	0	1

2.5.4.1.2 Binary BCH codes use Galios Fields by abstracted sequential symbols as polynomials and performing modulo-2 operations on them transforming them to code words. The remainder of this section will only concern itself with binary BCH codes and hence any reference to BCH codes this section refers to binary BCH codes. In BCH code applications each bit is a symbol. Hence a bit sequence '01001001' is represented as $v(\alpha) = \alpha + \alpha^4 + \alpha^7$. The size of the BCH Galio Field is $N = 2^m$, hence the code length of the encoded sequence is always equal to Equation 2.5.1.

$$n = 2^m - 1 \tag{2.5.1}$$

The polynomial used to generate the encoded sequence is known as the generator polynomial $g(x)$. The generator polynomial is constructed out of primitive polynomials $\phi(x)$. Primitive polynomials are polynomials, through which the entire Field can be generated [12]. The generator polynomial is the Lowest Common Multiple (LCM) of primitive polynomials, shown in Equation 2.5.2.

$$g(x) = LCM\{\phi_1(x), \phi_3(x), ..., \phi_{2t-1}(x)\} \tag{2.5.2}$$

Since each primitive polynomial increases the Hamming distance by 1 and the LCM is at most increased by m for each added primitive polynomial, Equation 2.5.3 expresses the number of parity bits added.

$$n - k \leq mt \tag{2.5.3}$$

The minimal Hamming distance between the various encoded blocks is the number of primitive polynomials used. Hence Equation 2.5.4 holds.

$$d \geq 2t + 1 \tag{2.5.4}$$

2.5.4.1.3 Encoding BCH Codes is a straightforward procedure. The design stage is the most extensive part of the process, but it is a once off effort. The actual encoding involves very simple binary operations, which effect long division.

In order to design a BCH code the desired error correction capability is required. This error correction capability shall be called t. The generator polynomial is created by Equation 2.5.2, using t primitive polynomials. The length of the encoded block size, Equation 2.5.1, is then deducted from the length of $g(x)$. This gives the length 'n' of the message signal.

The encoding process involves finding the parity-check bits, which let each of the primitive polynomials and the code block evaluate to zero. This is done by modulo-2 dividing the message by the generator polynomial. The remainder is then added to the message and the new block is the encoded block. Modulo-2 division can be done either in hardware or in software.

2.5.4.1.4 Decoding BCH Codes is a three step process. The first step is to calculate the syndrome values of the received vector. Next the error locator polynomials are calculated. Finally the roots of the error location polynomial provide, if 't' or less errors occured, the error locations.

Each syndrome of a block code is obtained by modulo-2 multiplying the received block by each primitive polynomial used in the creation of the generator polynomial. The error free block will cause all syndromes to evaluate to 0. Hence any non zero syndromes obtained are a direct result of corrupted bits. Any method of finding the specific combination of error positions (β in Equation 2.5.5) causing the set of obtained syndromes is a decoding algorithm. The Peterson-Gorenstein-Zierler algorithm and Berlekamp-Massey algorithm are examples of such algorithms.

$$S_n = \beta_1 + \beta_2 + ... + \beta_v \tag{2.5.5}$$

To aid the finding of the error locations a polynomial, the error location polynomial, is defined. The structure of the error location polynomial is shown in Equation 2.5.6. Finding the error location polynomial is the most challenging step, hence much research may be found on this matter. The result of the research is a host of algorithms, which simplify the finding of the error location polynomial.

$$\sigma(X) = (1 + \beta_1 X)(1 + \beta_2 X)...(1 + \beta_v X) \tag{2.5.6}$$

The final step is to find the roots of the error location polynomial, Equation 2.5.6. The roots are the inverse of the error location numbers. Hence the

error location numbers have been found and the corresponding bits may be flipped, correcting the errors.

2.5.4.2 Reed-Solomon

RS codes were invented by Irving S. Reed and Gustave Solomon in 1960. Reed-Solomon codes are a special subset of BCH codes. The information presented in this section was drawn from the book [13].

In the Section 2.5.4.1 binary BCH codes, their construction, encoding and decoding were discussed. Binary BCH codes, forming a subset of general BCH codes, abstract each bit in the transmitted message as a symbol. In contrast to binary BCH codes, RS codes group bits together to form code symbols. The symbols containing n bits has a length of $2^n - 1$ symbols (Equation 2.5.7). This makes the size of a RS code block even more rigid than binary BCH. The size of RS code blocks using byte size symbols is hence 255. RS codes using byte size symbols sometimes employ padding to shorten the code block. Padding is done by adding zeros to the front of the message and not transmitting these zeros. At the decoding end the same number of zeros are added.

$$n = 2^m - 1 \tag{2.5.7}$$

The number of error control bits is proportional to the error correction capability of the code (Equation 2.5.8) . This is in contrast to binary BCH codes where the number of redundant bits may be less than the upper bound placed by mt (Equation 2.5.3).

$$n - k = mt \tag{2.5.8}$$

The minimum Hamming distance of an RS code block is the same as that of any general BCH code. The distance is given by Equation 2.5.4.

The steps outlined for encoding and decoding binary BCH codes are the same for RS codes, with one addition. Knowing the error location of a symbol is enough to flip and hence correct the bit in error. In RS codes the added step is calculating the error values at the error locations.

The most important feature of RS codes is that they correct symbol errors, not bit errors. This means that $t + 1$ corrupted bits in independent symbols lead to corrupted data, but more than $t + 1$ consecutive corrupted bits can be corrected. This feature makes RS codes very good for correcting error bursts.

The discussion in this section is limited to the use of time domain encoding and decoding of BCH codes. A discussion of frequency domain methods to encode and decode BCH codes, including RS codes, may be found in [13].

2.5.4.3 LDPC

LDPC codes were invented by Gallager in 1963. The information presented in this section is taken from the book [14].

The original definition of LDPC is that the parity code occupies the null space of an (n-k)×n matrix H. In the original definition, the rows were designed to contain K ones and each column J ones. Since each one is contained in a row as well as a column, the baude rate is:

$$R = 1 - \frac{J}{K} \tag{2.5.9}$$

Each bit in the code is involved in J parity check sums. An incorrectly received bit will cause many of the check sums to fail. The reason why this method is called "low-density", is because of their usually small values of J and hence K.

An error is detected when parity checks fail. Bits are flipped and then the parities may be recalculated. Incorporating soft decisions makes LDPC an implementable FEC method.

2.5.4.4 Convolutional Codes

Convolutional codes were invented by Andrew Viterbi in the 1960s. The information presented in this section was taken from [14].

Convolutional codes form a subset of tree codes. Convolutional codes differ from block codes in that they make use of memory of previous states. For every k bit input symbol is converted to an n bit code. The n bit code symbol is a function of the current k bit input word as well as of the m previous input words. Convolutional codes are classified by their code rates and constraint lengths.

The time sequence of the code symbols forms a trellis. Usually a new input symbol can only cause a subset of the output symbols to be created. The encoder begins and ends at a known state.

The decoding of the convolutional code involves traversing the trellis and if every symbol follows plausibly from the preceding symbols then it may be assumed that the message was received error free. If an error occurs an unexpected symbol will appear in the trellis. The most likely path that the encoder took will then be chosen as the input.

In the presence of multiple errors the most likely path may not be the true path and hence much of the correct data will falsely be "corrected". Hence a very notable feature of convolutional codes is that if they receive an uncorrectable sequence an error burst is introduced.

Many algorithms exist for decoding convolutional codes. Of the most commonly used decoding algorithm for small values of k and n, is the Viterbi algorithm developed by Viterbi.

2.5.4.5 Concatenated Codes

Convolutional codes perform very well, with the exception of occasional error bursts. Since RS codes are very good at correcting burst errors, it compliments convolutional codes well. Combination of RS and convolutional codes are called concatenated codes.

The information data is RS encoded after which it is passed to a convolutional encoder. The inverse sequence of decoders is used. The output of the convolutional, usually Viterbi, decoder containing only burst errors is passed to the RS decoder.

2.5.4.6 Turbo Codes

Turbo codes were invented by Berrou, Glavieux, and Thitimajshima in 1993. These codes were the first implemented codes to approach the Shannon limit. The information presented in this section was taken from [13].

Turbo codes are encoded by separate different convolutional encoders. Transmitted is a block consisting of the original data, followed by the sub-blocks of encoded data parity pits.

When a corrupted code block is received, the convolutional decoders try and decode a path giving a message signal. If the decoded message signals of the decoders agree, it then means that the code block was correctly decoded. In the case that the decoders do not agree on the message, then the expected message code is exchanged along with the likelihood with which the message was obtained. This is repeated until agreement is found between the decoders.

2.5.5 Error Detection

ECC are investigated in the field of information theory. Assuring the correctness of data is an important aspect of computer systems as a whole. In communications it is desirable to know that the received data has not been corrupted. The two most commonly used ECCs are discussed in this section, these being parity checks and the Cyclic Redundancy Check (CRC).

2.5.5.1 Parity Check

The well known parity check appends either a high, or low, bit to data blocks, forming code blocks. The 1's, in each of these code blocks, either modulo-2 sum to 0, in even parity, or 1, in odd parity. Parity check introduces a Hamming distance of 1 between code blocks and enables single error detection. This simple check is prevalent in very low BER environments, such as embedded systems.

2.5.5.2 CRC

A CRC is a reliable error detection mechanism. A code word is encoded by a generator polynomial over a finite Galois field. The mechanics of this calculation is very similar to a BCH encoding process. The encoding using a CRC-16 code will result in a Hamming distance of 16. Hence all errors of up to 16 bits will be detected. Further more than 99.99% of all burst errors larger than 16 bits are detected. Furthermore only very few error combinations consisting of more than 16 errors transform one code word to another. This results in CRC being a very strong ECC method and is almost exclusively used to ensure data correctness when parity checks become insufficient.

2.6 Network Protocol Layering

When considering network protocols a natural presentation abstraction is to divide a protocol into layers. Each layer performs a well defined task. The next layer relies on the service of the lower layer. The layers are stacked producing a network protocol stack.

Each layer gives the appearance of end to end communication. In actual fact,, all layers divert the task of data transfer to the lower layer. The lowest layer is the only one that physically sends data between the computers.

In the next couple of sections a discussion on the commonly used Open System Interconnection (OSI) 7 layer model will follow. The Internet 4 layer model is also discussed in the following paragraphs.

2.6.1 IP 4 Layer Stack

In 1974 a paper was published presenting a huge protocol known as the Transmission Control Program [15]. The Transmission Control program was based on Packet Switching networks. Later this protocol was divided into modular layers, which eventually led to the TCP/IP stack and IPv4.

The IP is a 4 layer stack. It does not have all the layers present in the OSI 7 layer model. This discrepancy can be described to the purpose fact that the IP model preceded the OSI model. The 4 layers are shown in Figure 2.6.1.

The TCP/IP stack remains the most popular stack used in modern networks. Notwithstanding the popularity of the IP stack the following layer by layer discussion will refer to the more extensive OSI model.

2.6.2 OSI 7 Layer Model

The need for a standard governing the interconnecting between various computer networks was recognized by the International Standards Organization (ISO). During 1977 a committee was formed to formulate a standard and by

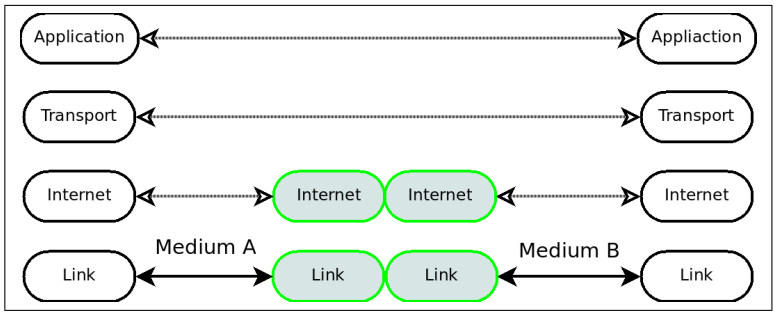


Figure 2.6.1: The 4 Layer IP Stack

1979 the OSI Model was formulated. The schematic presentation of the OSI Model may be found in Figure 2.6.2. [16]

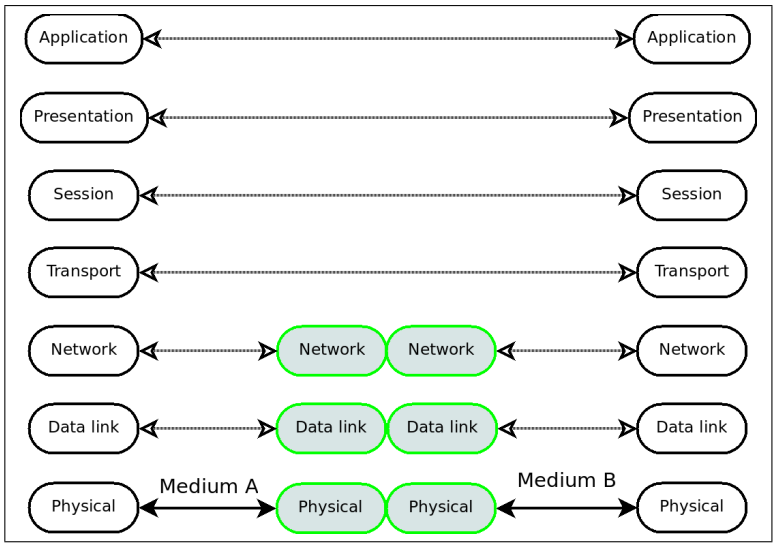


Figure 2.6.2: The 7 Layer Open Systems Interconnection Model

The OSI model was required to connect various proprietary networks. The proprietary network protocols of that time did not even use the same text encoding. This requirement is responsible for the detail of this model. Many of these layers are not present in the other protocol stacks.

Each layer present in the protocol stack adds a header to the payload data. This header is used in the processing of that layer.

2.6.2.1 Physical Layer

The physical layer is the first layer in the OSI model. The physical layer performs the service of transfering a data stream over a specific medium. The physical layer is the only layer, which actually transfers data, all other layers rely on the physical layer for communication.

A physical layer transmitting data has a physical bit stream as an input and an electrical or or electromagnetic signal as an output. A physical layer receiving has an electrical or electromagnetic signal as an input and a bit stream as an output.

In wireless links the bit stream is transfered over radio signals which are modulated using schemes discussed in Section 2.4.4. Methods such spread spectrum techniques and frequency hopping are also employed in wireless links.

2.6.2.2 Datalink Layer

The datalink layer is the second layer in the OSI model. In the IP stack there is no distinction between physical and datalink layer, hence many modern datalink protocols include physical layers in their specifications. The datalink layer performs the service of node to node connections, within a network.

A datalink layer protocol program transmitting data has length constrained data chunks as an input and a bit stream as an output. The datalink program at the receiving end performs the inverse. The datalink layer typically adds a header to the payload data which is used for administrative purposes.

The datalink protocols discussed in this section have been classified according to the distance they were designed for. Networks can be classified as one of the following depending on their reach: Personal Area Network(PAN), Local Area Network (LAN), Metropolitan Area Network (MAN) and Wide Area Networks (WAN).

In Wireless PAN (WPAN) the 802.15, Bluetooth, protocol is the most prevalent. The most common examples of Wireless LAN (WLAN) are the IEEE 802.11 family. The IEEE 802.11 also features in Wirelss MAN (WMAN) networks, equally prominent are the IEEE 802.16 and IEEE 802.20 protocols. Wireless WAN (WWAN) are characterised by GSM, GPRS, 3G as well as satellite protocols. Satellite specific datalink protocols will be discussed in further detail in Section 2.6.3.

Due to the high cost of a physical medium infrastructure, theses mediums are often shared. Sharing a medium creates the possibility for multiple users attempting to access the medium simultaneously. A design feature of many datalink layers is the Medium Access Control mechanism, which governs the access of the physical medium.

2.6.2.3 Network Layer

The network layer is the third layer in the OSI model. The task which protocols on this layer perform is that of end-to-end delivery of packets through one or more networks. The sending end as well as the receiving end are uniquely identifiable.

A network layer protocol program transmitting data has length constrained data as an input and a packet or frame as an output. The network layer program at the receiving end performs the inverse operation. All following layers may be described in this way each adding a header which contains information with which the receiving end may perform its task.

The most commonly used network protocol is the IPv4 and soon IPv6. Space specific network protocols are the Space Packet Protocol and Encapsulation Service [17].

2.6.2.4 Transport Layer

The transport layer is the fourth layer in the OSI model. Protocols at this layer provide a host to host transport medium, making a distinction between various users and programs desiring to use the network infrastructure.

The two most used session layer protocols are Transmission Control Protocol (TCP) and User Datagram Protocol (UPD). Used for reliable and unreliable data transfer over an IP network.

2.6.2.5 Session Layer

The session layer is the fifth layer in the OSI model. Protocols at this layer provide the a session long communication link between network computers. A session is usually established when requested by an application.

Authorisation protocols usually occupy this layer. Programs include Secure SHell (SSH), Secure CoPy (SCP) and Password Authentication Protocol (PAP). These programs require the same link to be used throughout a session. At the start of the session authentication required.

2.6.2.6 Presentation Layer

The presentation layer is the sixth layer of the OSI model. Protocols at this layer change data from a specific structure to a general structure and then back again. This layer is largely legacy, but the presentation of eXtensible Markup Language (XML) may be considered a presentation layer process.

2.6.2.7 Application Layer

The application layer is the seventh layer of the OSI model. The application is the actual program requiring the network service. Programs at this layer are diverse.

An example of a protocol program at this layer is the File Transfer Protocol (FTP). FTP is a program that assures the correct transfer of files between computers on an IP network.

2.6.3 Space Network Specific Datalink Protocols

Considering the popularity of the TCP/IP stack in modern computer systems, it would be a natural jump to assume that the TCP/IP stack, in conjunction with the FTP, should be employed in the protocol architecture. Due to, among other factors, the inherently long delays in space datalink protocols the TCP/IP stack is precluded form space networks [17]. Previous attempts of using standard IP protocol stacks have included the use of Advanced Trivial File Transfer Protocol (ATFTP), a variation on Trivial File Transfer Protocol (TFTP), over a UDP/IP stack [18]. TFTP is much less efficient than FTP [19], which has built-in retardation mechanisms to assure the network is not flooded [20].

In the previous paragraph, the use of space specific datalink protocols was mentioned. The questions to be asked, are why use such protocols and how do they work. Such use is required by the high BER that space links are subjected to and also to account for the high propagation time. Both these factors have been considered and designed for in the various protocols. A further benefit to using a space specific datalink protocol is a somewhat anecdotal desire to use a protocol that has been "tried and tested".

The Green Book (Informational Report) [21] gives an overview of various space specific and other protocols and how these should be used in the protocol design. Figure 2.6.3, taken from the Green Book, gives a schematic overview of how network protocol stacks may be combined.

The remainder of this section discusses the following datalink protocols in more detail:

- Telemetry Space Data Link Protocol (TM-SDLP)
- Telecommand Space Data Link Protocol (TC-SDLP)
- Advanced Orbiting System Space Data Link Protocol (AOS-SDLP)
- The Proximity-1 Protocol suit

Complementary versions of these space datalink protocols have been defined by both the CCSDS as well as ECSS and the respective definitions often vary only slightly. None the less, the ECSS suit of standards have been used and where discrepancies exist, the definitions in the ECSS standard prevails. The datalink layer has been divided into sublayers to cater for the added requirements which do not appear in ground based protocols. The lower one is the Synchronisation and Channel Coding Sublayer (SCCS) and the higher

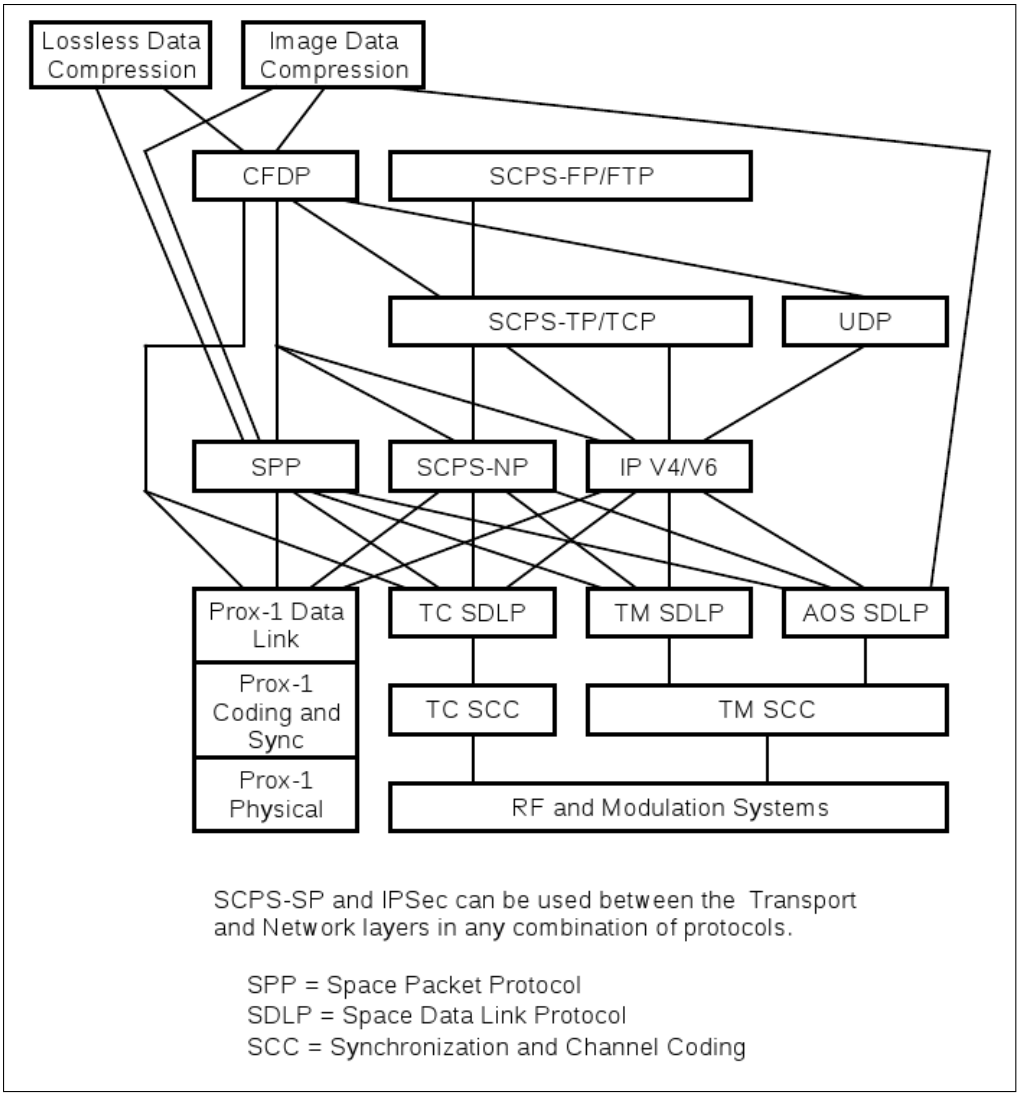


Figure 2.6.3: Various Space Protocol Network Stack Setups

is the Transfer Sublayer (TS). In CCSDS literature the upper sublayer is referred to as Datalink Protocol Sublayer. The TC-SDLP has a further mandatory additional sublayer, the SS, which is optional in other protocols.

Each of the sublayers has a specific task to fulfil. The purpose of the SCCS is to add FEC onto the data and to make sure that the receiver remains in time with the transmitter. The purpose of the TS is to transmit data frames between two nodes. The purpose of the SS is to both combine short packets from various endpoints as well as to segment too long packets. Note that the words "packet" and "frame" are used interchangeably in this document. This is due to the consistent use of the word "frame" for a variable length data unit in the ECSS documents, whereas such a data unit is commonly referred to as a "packet" in other literature including some of the CCSDS documents.

2.6.3.1 TM-SDLP

Typical uses for the TeleMetry - SDLP (TM-SDLP) include sending telemetry data. The TM-SDLP offers a best effort link between two nodes in the network. The TM-SDLP may be configured to run synchronously, periodically or asynchronously depending on mission requirements. The TM-SDLP has two mandatory sublayers, the TeleMetry - TS (TM-TS) and the TeleMetry - SCCS (TM-SCCS) which will be discussed in the following paragraphs.

2.6.3.1.1 The TM-TS establishes a one way (monodirectional) link between a sending space/ground node and a receiving space/ground node. The TM-TS transmits fixed length frames, the size of which is set prior to the mission. During transmission the protocol maintains frame sequence, but both loss and duplication are possible. Data, which is smaller than the length of a frame, is concatenated into a single frame. Data longer than a frame is split into multiple frames. The TM-TS provides Virtual Channels (VC) for various higher layer services to access the same physical channel. [22]

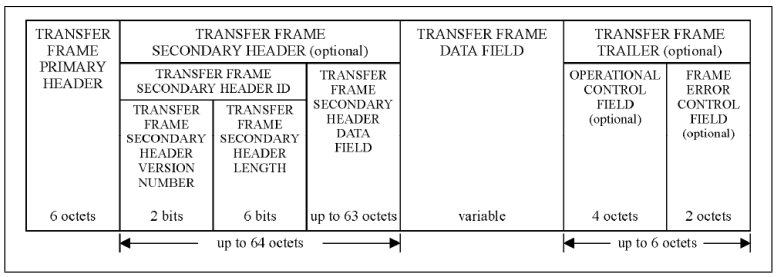


Figure 2.6.4: Telemetry Transfer Frame Format

The VC of the TM-TS is equivalent to the Logical Link provided by the Logical Link Control Layer (LLC) of datalink protocols for WLANs. The

concept of VCs is more pronounced though, than in ground networks. The reason for this is that unlike most terrestrial networks, which exclusively use IP over the datalink layer, spacecraft specific data exchange is necessary. Such exchange includes clock calibration and reporting data.

The variable length user data packet is placed into a TM-TS Frame. If the user packet is shorter than the available space in the frame, then the next user packet is added, if there is no more user data then filler data is appended. However, if the user packet size of any of the packets is greater than the space available in the TM-TS Frame, the packet is split and the remainder is placed into the next TM-TS Frame. In the event of no user data being available, the data field is filled with filler bits. This is done to not lose synchronisation.

The frame structure of a TM-TS Frame is shown in Figure 2.6.4. An optional CRC may be added to the TM-TS Frame, discarding a corrupted frame. Once the packet has been framed, it is passed to the TM-SCCS.

2.6.3.1.2 The TM-SCCS receives data frames, usually either TM-TS Frames or Advanced Orbiting System - TS (AOS-TS) Frames, and sends them over the radio link. At the TM-SCCS FEC is added to the data to improve the link quality. The TM-SCCS assures the bit synchronisation is maintained. [23]

The FEC mechanisms that have been suggested in the standard are RS -, convolutional -, concatenated codes as well as Turbo Codes. RS codes represent bytes as symbols and up to n erroneous symbols can be corrected in the code word. This property makes RS codes apt at correcting burst errors. Convolutional codes are decoded using a maximum likelihood decoder. The most likely path is usually the correct one, therefore, if an error is falsely decoded a burst of errors occurs. Concatenated codes use the strength of both FEC schemes. The output of the convolutional decoder is decoded using a RS decoder. Turbo Codes are very strong FEC which have very low code rates these are typically used for very poor BER. The aforementioned FEC have been specified in the standard [23]. The FEC defined in this standard may be replaced by a mission specific FEC design.

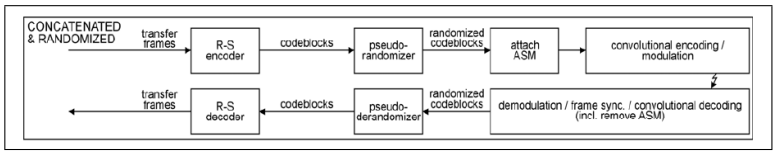


Figure 2.6.5: Telemetry Synchronisation and Channel Coding Sublayer with Concatenated Code

In order to maintain synchronisation a synchronisation marker is transmitted prior to data. This marker is known as the Attached Sync Marker (ASM). The ASM, usually a 4 byte sequence, which when detected indicates

the beginning of data. The following data is then decoded and the decoded frame is transmitted to the higher sublayer. An example setup with Concatenated codes is shown in Figure 2.6.5.

If a large amount of successive logical high or low are received the timing may start drifting, resulting in bits being lost or added. A high amount of bit transition is desired, and may be achieved by pseudo randomising the data prior to transmission. A pseudo randomisation algorithm is defined in the standard. To see where in the setup this is done, reference is made to Figure 2.6.5.

2.6.3.2 AOS-SDLP

The Advanced Orbiting System - SDLP (AOS-SDLP) was added to the space datalink layer suit when the International Space Station required additional data to be transfered. Data such as voice and video type on-line data needed to be transmitted concurrently with the traditional "telemetry type" - and "telecommand type" data. A bidirectional link is created by the AOS-SDLP which can transport any class of data. [17]

The AOS-SDLP is divided into two sublayers. The lower sublayer is the TM-SCCS, which has been discussed in Section 2.6.3.1. The upper sublayer, AOS-TS, is defined in the document [24].

The AOS-TS makes use of fixed length frames. The structure of a frame is shown in Figure 2.6.6. The AOS-TS makes use of VC as described in Section 2.6.3.1. Unlike the TM-SDLP, AOS-SDLP is not limited to the receiving packets of any specific nature from the upper layer. Even bit streams are accepted and packaged.

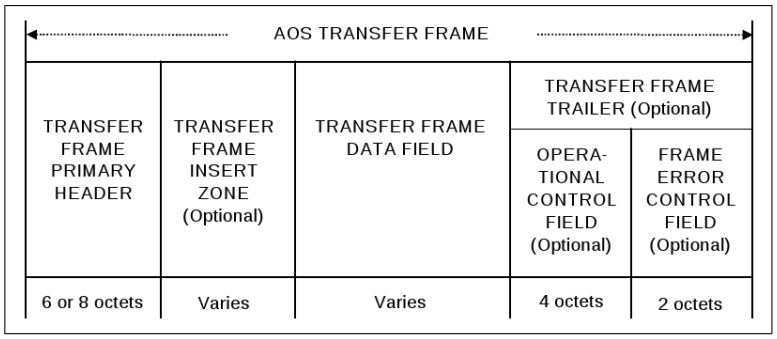


Figure 2.6.6: Advanced Orbit System Transfer Frame Format

The logical organisation of payload data coming from the upper layers is shown in Figure 2.6.7. Various functions have been defined in the AOS-TS standard to manage the various incoming data types at the sending end. The details of each of the functions may be found in [24].

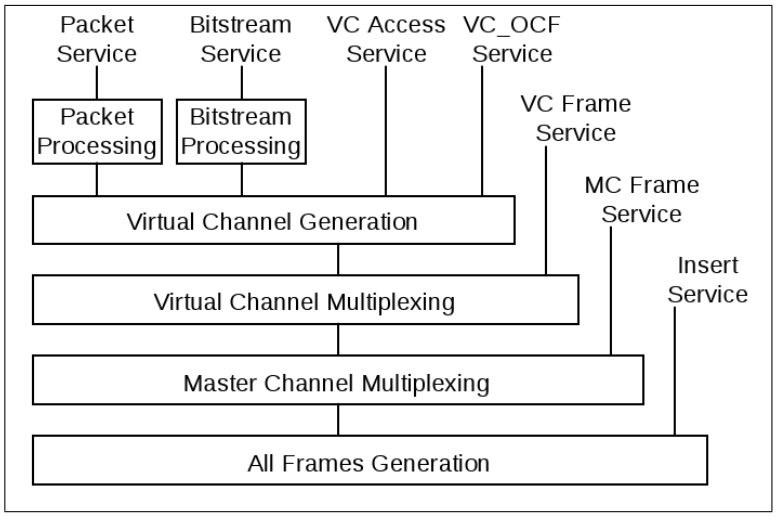


Figure 2.6.7: Logical Organisation of Sending End Functions

The AOS-SDLP is an upgrade of the TM-SDLP. AOS-TS frames carry as version 2 identifier a binary "01", as opposed to TM-TS which carry binary "00". The use of the existing TM-SCCS lets space link designers reuse working FEC mechanisms.

2.6.3.3 TC-SDLP

Typical uses for the TC-SDLP include sending commands over space links. The rationale behind the TC-SDLP is reliability. A command should be received correctly and in sequence. The TC-SDLP was designed to meet these requirements. [17]

The TC-SDLP consists of three sublayers. The highest of these layers is the TC-SS, next the TC-TS and finally the TeleCommand - SCCS (TC-SCCS). Figure 2.6.8 shows the various sublayers and their interactions. Each of these sublayers will be explored in further detail in the following paragraphs. [25]

2.6.3.3.1 The TC-SS receives payload data from a higher layer and splits, or assembles, it into segments. These segments are then passed to the lower TC-TS. The TC-SS may multiplex user data from various VC as described in Section 2.6.3.1. The segments have a primitive sequencing protocol, which identifies four types of segments. These four types are:

- A first segment is identified by a binary '01'. Such a segment implies that user data has been segmented and this segment is the first of these segments.

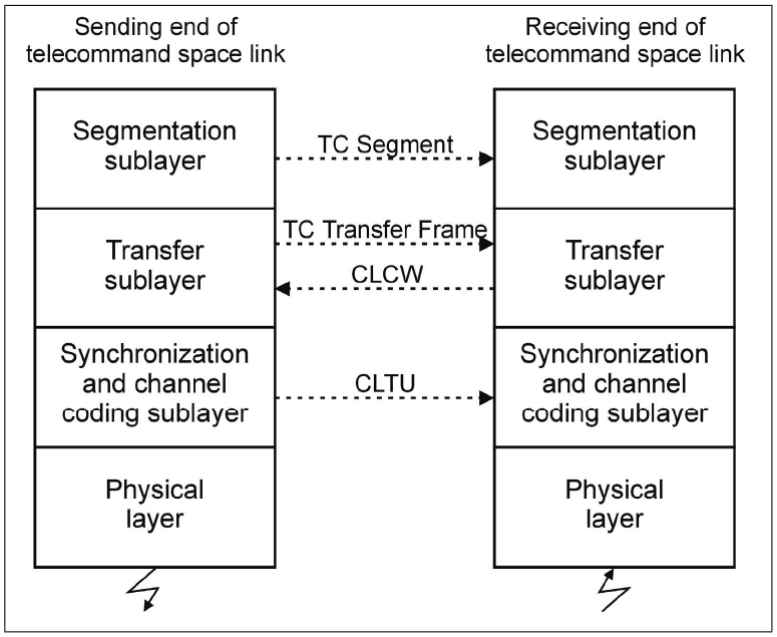


Figure 2.6.8: Telecommand Datalink Layers Sublayers

- A continuing segment is identified by a binary '00'. A continuing segment is a segment of user data.
- A final segment is identified by a binary '10'. This is the last segment in a sequence of segmented user data.
- A no segmentation segment is identified by a binary '11'. Such a segment implies that the user data has not been segmented.

The TC-SS sequencing mechanisms does not offer sequence numbering and as such does little more than identify underlying server errors. If an illogical sequence of segments is received, the link may be aborted. The next lower sublayer, the TC-TS, provides the reliability and therefore under normal operations no illogical sequences should occur.

2.6.3.3.2 The TC-TS contains the mechanisms responsible for reliable and in sequence data transfer. The TC-TS at the sending end of the space link transmits TC-TS variable length frames, with a maximum length of 1024 Bytes. The structure of TC-TS Frames is shown in Figure 2.6.9. Each of these frames contain a unique sequencing number. The TC-TS at the receiving end of the space link transmits acknowledgement (ACK)s called Communications Link Control Word (CLCW). The CLCW contain a sequence number, which corresponds with the next expected TC-TS Frame. The procedure of acknowledging arrived frames is known as a sliding window protocol. The specific

Sliding Window variant used in this standard is known as "Go-Back-N". The alternative sliding window protocol is Selective-Reject. More detail on this may be found in [26].

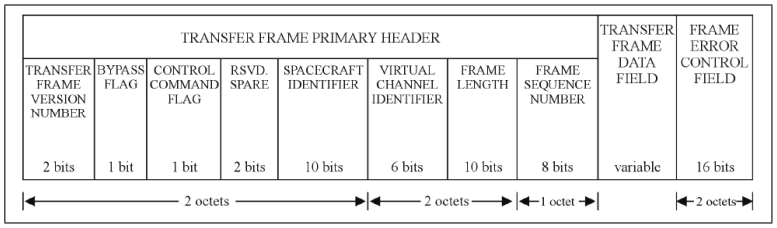


Figure 2.6.9: Telecommand Transfer Frame Structure

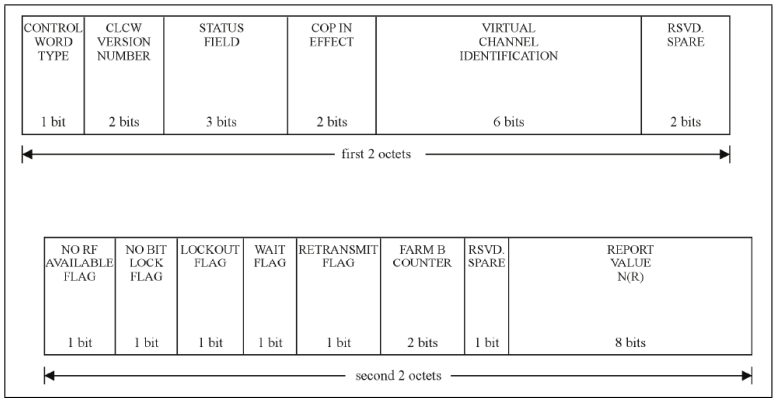


Figure 2.6.10: Telecommand CLCW Structure

The TC-TS also offers a unacknowledged service. This allows the transmission of unacknowledged frames. These frames are called bypass (type-B) frames.

The Communications Operation Procedure (COP-1) runs in the TC-TS with two distinctive processes running at the sending and receiving end respectively. At the sending end the Frame Operation Procedure (FOP-1) maintains all internal states. At the receiving end the Frame Acceptance and Reporting Mechanism (FARM-1) maintains all internal states. The detailed workings of COP-1 and FARM-1 are quite complex and more detail on this and other aspects of the TC-SDLP may be found in Chapter 4.

TC-TS contain a CRC check sum. This check sum causes frames with residual errors to be dropped. TC-TS frames are passed down to the TC-SCCS.

2.6.3.3.3 The TC-SCCS offers the services of maintaining synchronisation between sending and receiving ends of the space link as well as adding FEC to improve link quality. Frames, which are sent over the TC-SCCS, do not have a set length. The frame is split up into very small cells of length 56 bits. The use of small cells to transport data over a physical link is also be found in the Asynchronous Transfer Mode (ATM) standard [27]. Dividing data into cells has the advantage of decreasing the chance of any such cell of failing.

In the TC-SCCS BCH codes are added to the 56 bits data to correct any single error that may have occured in the cell. A BCH code (62,56) is used for this purpose. Additional error detection capability is added onto this data by adding a parity check. This results in a 63 bit code word. A filler bit is appended to the code word giving an 8 byte cell, which is passed to the physical layer to transmit. The receiver can decode any code word with a single error. Any code word with a double error will be identified and is rejected.

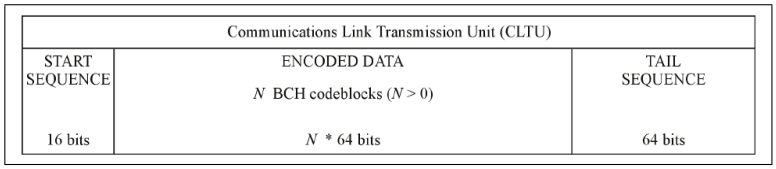


Figure 2.6.11: Telecommand Synchronisation and Channel Coding Sublayer Transmission Unit Structure

A data frame is preceded by a start sequence marker and concluded by a tail sequence marker, this is shown in Figure 2.6.11. The receiving end searches for the start sequence. This is accepted, even with any single error present. The TC-SCCS then continues to decode the cells until an undecodable cell appears. The tail sequence is an undecodable cell. The searching state is entered again after the tail sequence is received.

As was the case with the TM-SCCS the data is pseudo randomized. The pseudo randomisation insures maximum bit transitions.

2.6.3.4 Proximity-1

The Proximity-1 protocol suit is typically used for close range space communication. The Proximity-1 Synchronization and Channel Coding Sublayer (Proximity-1-SCCS) does not support error correction, but it does specify the use of error detection. The rationale behind the protocol Proximity-1 protocol suit is to maximise the often time constrained communications opportunities. The premise of a good link is underlying to the use of the Proximity-1 protocol suited. A good link may usually be expected at short distances, since

tracking antennas follow spacecrafts at LEO and MEO and the lack of atmospheric attenuation between space craft and ground. [23]

The Proximity-1 protocol suit defines the entire sub-network layer in OSI terms. It defines the datalink as well as the physical layers. The datalink layer is divided into the Proximity-1 Transfer Sublayer (Proximity-1-TS) and the Proximity-1 Synchronisation and Channel Coding Sublayer (Proximity-1-SCCS). Figure 2.6.12 gives an overview of the Proximity-1 protocol suit. [28]

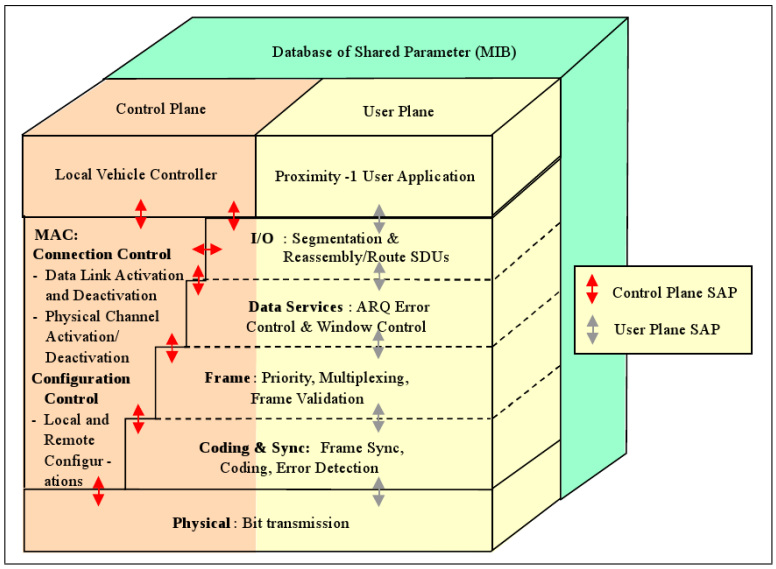


Figure 2.6.12: Proximity-1 Layered Architecture

2.6.3.4.1 The Proximity-1-TS ensures in sequence data transfer without loss or duplication. The Proximity-1-TS offers the services of framing, media access, data services, and input-output. Each of these services are contained in a logical unit. The placing of each of these logical units is shown in Figure 2.6.12. [29]

The sequencing is accomplished using a variant of the Communications Operation Procedure (COP-P). This mechanism accomplishes the task of sequencing by identifying duplicate frames and notifying about and resending lost frames. The operations are very similar to that of the TC-SDLP COP-1.

The MAC service establishes and ends communication sessions. The configuration is controlled of the physical layer, both local and remote. Due to the possibility of corruption of data at the physical layer, configuration services are persistent and require handshaking.

The input-output service is performed at the interface between Proximity-1 and the next higher protocol layer. The next higher protocol layer specifies

the desired quality of service as well as other session specific data. After a session has been initiated, data is passed to Proximity-1-TS by the input-output service.

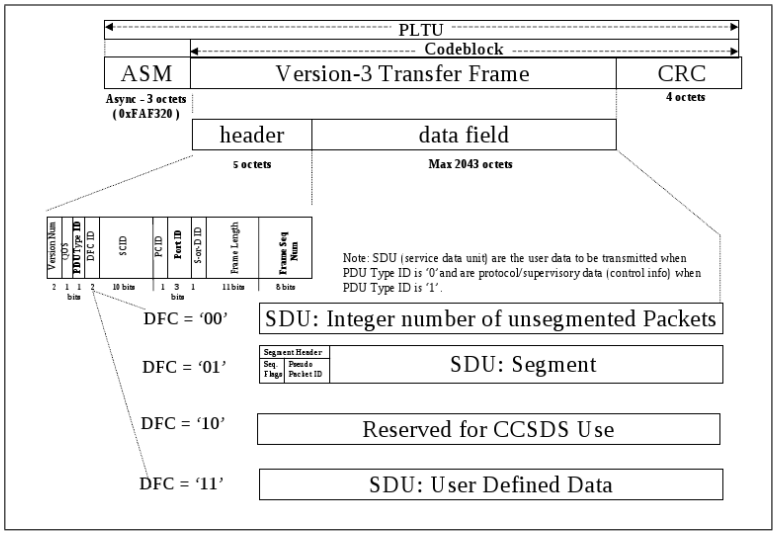


Figure 2.6.13: Proximity-1 Layered Architecture

The framing service is responsible for placing the user data into frames. The frame structure is shown in Figure 2.6.13.

2.6.3.4.2 The Proximity-1-SCCS provides the service of error detection and synchronisation. FEC in the form of Convolutional codes may be added at the Proximity-1-SCCS. [30]

A synchronisation marker is added to the beginning of a frame and a CRC is appended to the end of a frame (Figure 2.6.13). Synchronisation may be lost if no signal is received. Such a synchronisation loss typically occurs if the frames are sparse. The Proximity-1-SCCS specifies the use of an idle signal if no frames are ready for transmission. The CRC rejects corrupted frames.

2.6.3.4.3 The Proximity-1 physical layer is defined in [31]. Frequencies of operation in the UHF band have been specified, but use of other bands has not been excluded. The signal is specified to be Bi-Phase-L encoded. This encoding mechanism is also known as Manchester Encoding and used on common Ethernet [32].

2.7 Protocol Layering as Applicable to the MCSP

The MCSP is required to transfer files between remote ground stations. The individual ground stations may be stand alone computers or they may be part of a local network, but these networks are not interconnected. Hence there is only one ground station per ground station network.

The satellite passes over the ground station several times a day, but communication will not necessarily be initiated during every pass. The rest of the ground station network is connected continually. The link between the ground stations and the satellite is the bottleneck of the Multichannel payload system.

A bottleneck should at all times work at full capacity. Therefore the transferring of data files between the ground station networks should not at any point interfere with the satellite ground station link. Thus any data files which need to be transferred by the satellite will have to be stored at the specific transmitting ground station until communication has been initiated by the satellite and a transfer may be attempted. A data file should also only be transferred to the rest of the ground station network once it has been successfully received.

2.7.1 Optimising the satellite ground station link

The satellite ground station link has been shown to be the bottleneck of the system and should be optimised. The ground station network protocol stack will be discussed, after which the ground station satellite network protocol stack will be discussed.

2.7.1.1 Ground station network protocol stack

The ground station network is not at all limited to size or layout. The number of nodes may be numerous, routers may well be present and diverse physical layer mediums may be used throughout.

A computer or user in any ground station network will transfer files to the network's ground station. This file will have to be accompanied with another file containing information on:

- The final destination
- The receiving ground station

This second file will hence forth be referred to as the header file.

Since the TCP/IP is the most used protocol stack to employ in current network systems with readily available implementations of most standards, this is definitely the best choice for the ground station system. The protocol design is given in Figure 2.7.1.

The application layer of the ground station network will have to be designed to create a header file alongside the data file to send to the ground station. Such a management program can then make use of FTP to transfer the files.

2.7.1.2 Ground station satellite protocol stack

When data files and header files are transferred between a ground station and the satellite, this transfer only involves one single hop. A hop in the network protocol stack is usually fully managed by the data link layer.

The network layer and session layers both only introduce additional overhead and no additional gain. The only layer that is important, is the application layer where an FTP should be used.

The reason for the typical TCP/IP setup, over which FTP runs, is that the TCP/IP setup creates a reliable, in sequence transfer of data packets.

A datalink layer, which offers an in-sequence, reliable packet stream, in conjunction with an FTP protocol, would therefore be all that is required in the ground station satellite link.

In Section 2.6.3.3, the TC-SDLP protocol was introduced. The TC-SDLP offers an in-sequence, reliable data packet stream. The Proximity-1 protocol, Section 2.6.3.4 also offers such a service. Proximity-1 only offers Convolutional FEC, which has a code overhead of 50 % and more. TC-SDLP on the other hand makes use of BCH codes, which have a coding overhead of 12.5 %. Furthermore, the use of Manchester encoding versus pseudo randomisation halves the effective payload data throughput again. The TC-SDLP is hence the protocol of choice.

In the literature the CCSDS File Delivery Protocol (CFDP), a variation on the common FTP, running over a TC-SDLP setup is proposed in [21]. The CFDP is a feature rich protocol and a simplified version of this protocol may better serve the needs of this specific project. Since the CFDP and the file scheduling problem are an integral part of the application layer, the tailoring and implementation of this protocol falls outside the scope of this thesis.

The layered system protocol design is given in Figure 2.7.1. A note should be made about the sequencing of the transfers that take place. A data file is only transfered between satellite and ground station, once it has fully arrived at the sending node.

2.8 Throughput Considerations

In Section 2.7.1.2 the ground station satellite link was found to be the system's bottleneck. This section will discuss the performance parameters, goodput and effective goodput. Effective goodput is also refered to as effective data throughput.

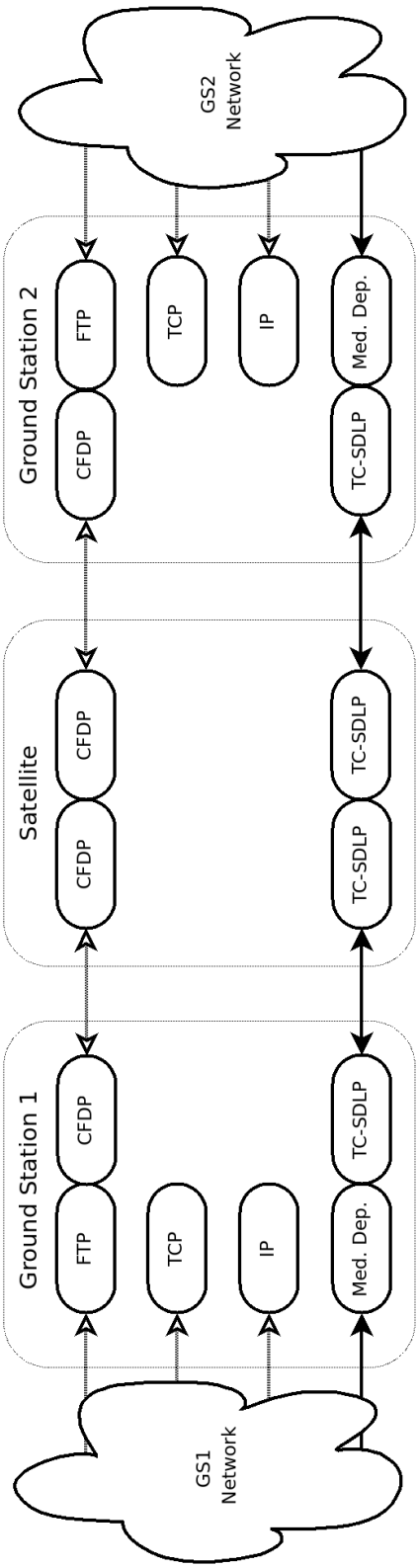


Figure 2.7.1: The Layers of the Multichannel Payload Satellite

2.8.1 Goodput

In an effort to maximise throughput the a performance measurement termed goodput was created. Goodput describes the perceived payload baud rate. The perceived baud rate is limited by the system baud rate, less the baud rate used by the underlying protocol layers.

Wireless links are much more error prone than modern wired links. Losses and errors occuring in wired networks are often ascribed to congestion induced -, rather than error induced packet loss. Hence in the literature goodput has been used to define optimal wireless link parameters. Some of the efforts that have been made were to optimise protocols such as the 802.11.a and 802.11b by changing parameters depending on the current goodput. The work includes analytical and empirical results [33] [34] [35].

The goodput may be taken at the TCP level, since this is truly how the user perceives the goodput. In [36] A single link TCP connection over a wireless link was investigated. In this study the notion of trading bandwidth for BER, to maximise goodput of TCP, is discussed.

The aim of most goodput studies was to find some sort of optimal set of parameters for a specific, or a set of, or any given quality; thus optimising the goodput of this link. The biggest problem with the notion of optimising a link for a given set of parameters, is that local optimisation does not mean network wide optimisation in large networks.

2.8.2 Effective Goodput

Effective goodput, or effective data throughput, quantifies the amount of data transfered normalized to the total data transfered. This quantity is independent of the system baud rate and as such it gives a method of comparing protocol efficiency regardless of the underlying system baud rate. In Section 3.3 effective data throughput is used, rather than goodput, as a quantity of measure. Using effective data throughput makes the results agnostic of aspects such as the baud rate, hence making the results generic.

2.9 Conclusion

The discussion in this chapter has surveyed some of the work done by previous researchers. Two of the major players in the space industry were introduced as well as two major space standardization bodies. Some of the standards offered by these bodies were presented.

The orbital dynamics of the satellite were discussed as a framework to understand the physical orbit and some of the orbital terminology. The factors impacting on signal quality and the introduction of noise was discussed. These factors included the modulation scheme and hence a superficial discussion of this was included. The BER resulting from this may require error

checking mechanisms or even FEC. Common error checking methods and many FEC methods were investigated.

The OSI model as well as the internet model were introduced. The OSI model was used as a framework for the discussion of protocols on various links. Some space specific protocols developed by the ESA and the CCSDS were presented. The datalink protocols, being the essence of the work presented in this report enjoyed a detailed discussion. A protocol stack choice flowed from the discussion of the datalink protocols and was presented.

The terms goodput and effective data throughput, as link measurement quantities, were presented to give a unit of measure. The core concepts have been introduced and the reader has been prepared for the the detailed system design presented in Chapter 3. In Chapter 3 many of the concepts introduced in this chapter are applied.

Chapter 3

System Design

In Chapter 2, the background study was presented. The study offered a background of previous work done in the fields pertaining to the MCSP and concluded with a suggested system layout.

This chapter investigates the application of engineering knowledge to the objectives stated in Chapter 1. The link budget is calculated, the specific problem environment discussed, and engineering solutions offered to these. Furthermore, optimisations are investigated and design choices presented. This chapter is concluded by a discussion of the design of a simulation model that simulates the real world implementation. The model will be used to predict and benchmark the system implementation. In Chapter 4, the implementation of the system is discussed. Results of the simulation, as well as the actual system, are presented in Chapter 5.

3.1 Tools used in the System Design

To understand and design a system, it needs to be simplified. This enables the identification and isolation of the particular aspects which should be measured and from which conclusions and optimisations can be made. The main tools used, were the Python and C++ programing languages. These two programing languages have different strengths and were used to solve complementary problems.

Python is an open source, free of charge, high level, object-orientated, interpreted programing language with extensive functionality and libraries [37] [38] [39]. Python was extensively used for its scientific and mathematical capabilities. Most of the calculations, as well as much of the data processing and manipulation presented in this chapter, as well as Chapter 5, were performed using Python as a tool. Some time insensitive simulations were conducted using Python.

C++ is a very complete and complex programing language. C++ has an object orientation layer which gives a clear oversight and improves code ma-

nageability and code re-usability. It also has access to low level and fast executing C. Hence, C++ is a great choice to combine manageability with speed. The trade-off lies in the complexity and restrictiveness of the language. This programing language was used to build the simulation model.

3.2 Link Budget Calculation

This section concerns itself with the link budget calculations. The link budget calculation gives a link quality in terms of the SNR, which is the link quality as perceived by the physical layer radio. From the SNR, the BER can be calculated. The BER gives a quantitative value of the link quality as perceived by the datalink layer. Hence, the BER is a basis from which FEC calculations and throughput considerations could be made.

The various factors pertaining to the link quality are named, and as far as possible, values are assigned to them. The background presented in Section 2.4 is extensively drawn upon.

3.2.1 Geometry

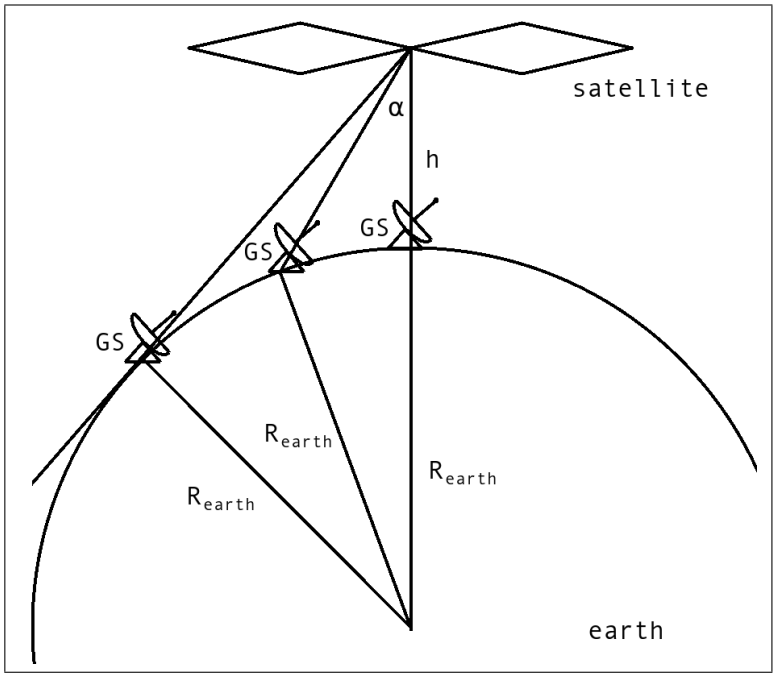


Figure 3.2.1: Ground Stations in Satellite Footprint

The geometry of the GS and satellite affect both their orientation toward each other as well as the distance between each other. The geometry is depicted in Figure 3.2.1. The variables and constants described in Section 3.2.1, are used throughout Section 3.2.

The satellite faces toward the centre of the earth. The distance between the centre of the earth and satellite is the sum of the earth's radius (R_{earth}) and the altitude of the satellite(h). Both these values were simplified to constants, R_{earth} to the average radius of the earth and h to 500 km.

The satellite, GS and the centre of the earth form a triangle (omitting rare alignments). Communication can only occur while the earth does not eclipse the satellite. From the perspective of the GS the satellite comes up over the horizon, hence the GS has a maximum orientation to the satellite (α_{GS}) of 90°. From the perspective of the satellite the GS enters the satellite's footprint. The satellite's maximum orientation toward the GS (α_{Sat}) is 67.997°, by Equation 3.2.1. When the GS comes into the satellite's footprint, the maximum distance between the satellite and the ground station (D) is reached, the value is 2573 km, calculated by Equation 3.2.4.1.

3.2.2 Orientation

The area on the ground which is within view of a satellite is said to be within the satellite's footprint. The satellite's maximum footprint is determined by the height of the satellite above the ground as well as the radius of the earth. How much of this footprint will result in successful communication depends entirely on the net effective BER.

As depicted in Figure 3.2.1, the GS in the footprint, the satellite and the centre of the earth form a triangle. The radius of the earth is known, as well as the altitude the of the satellite's orbit. Hence, the satellite's orientation toward a GS can be determined by the orientation of that GS toward the satellite. This calculation is done by the sine rule and is given in Equation 3.2.1.

$$\alpha_{Sat} = arcsin\left(\left(\frac{R_{earth}}{R_{earth} + h_{sat}}\right) sin(90^\circ + \alpha_{GS})\right)^\circ \tag{3.2.1}$$

3.2.3 Antenna

The GS and Satellite antenna are described in Sections 3.2.3.1 and 3.2.3.2.

3.2.3.1 Satellite Antenna

The SSIS S-band CTRS antenna, version QM, was designed for the satellite by Prof. KD Palmer of Stellenbosch University. During each successive pass in which GS's communicate with the satellite, the orientation of the satellite's antenna relative to the GS will vary. This means that the antenna's gain will

vary during the same pass depending on the orientation between the satellite and the candidate ground station.

The satellite antenna gain has been provided in the antenna's data sheet. For convenience purposes, the gain was approximated to a simple mathematical expression, as given in Equation 3.2.2.

$$Gain_{SatAntenna} = \begin{cases} 0.2\ \frac{dB}{1^\circ}(\alpha_{Sat}) + 11\ dB & > 20^\circ \\ 7\ dB & \leq 20^\circ \end{cases} \tag{3.2.2}$$

3.2.3.2 Ground Station Antenna

The ground stations are low cost remote types and the antenna of such a remote rural ground station will not track the satellite. A mechanism capable of tracking would cause an unacceptable increase in startup cost as well as undesirable maintenance. Hence the antenna cannot have a very high gain, since this would imply that the antenna would also be very directional, which would cause too much potential communication time to be lost.

There are two types of antennas that are typical candidates for purposes such as the non-tracking Ground Station antenna. The first is a Helix and the second a Quadrature Helix antenna. The Helix antenna, as opposed to the Quadrature Helix antenna, has higher gain with a narrower wide angle, but it would still be capable of covering most of one quadrant. Using a Helix, the GS antenna would be pointed in one quadrant, improving those communication opportunities, which happen to fall in that quadrant. The Quadrature Helix antenna has a doughnut-shaped radiation pattern. A Quadrature Helix GS antenna could be pointing upward, hence covering all quadrants, but providing less gain than the Helix antenna. The trade-off is between moderate gain on all, or a high gain on approximately half of the satellite's passes. At completion of this work, no such design decision had been made. To keep the signal pattern unaffected by assumptions, an omnidirectional antenna was used in the calculations.

3.2.4 Free Space Loss

The free space loss is a function of the distance between satellite and GS as well as the transmit frequency. Equation 2.4.1 has been simplified to the specific requirements of this project, resulting in Equation 3.2.3. As the satellite orbits, the distance between the ground station and the satellite varies. This, in turn, causes a varying free space loss.

$$L_{FS} = 20log_{10}(D) + 101.98 \tag{3.2.3}$$

3.2.4.1 Satellite Distance

Referring to Figure 3.2.1, it becomes evident that the distance between the satellite and the Ground Station can be determined by the cosine rule. The distance between the satellite and the ground station is a function of the satellite's orientation. The relationship between the orientation of the satellite and the distance is given in Equation 3.2.4.

$$D = (R_{earth} + h_{sat})cos((\alpha_{Sat})) - \sqrt{R_{earth}^2 - ((R_{earth} + h_{sat})sin(\alpha_{Sat}))^2} \quad (3.2.4)$$

Section 3.2.2 shows that the satellites's orientation is a function of the GS's orientation and can be determined by Equation 3.2.1. This Equation could be rewritten in terms of the GS orientation, by Equation 3.2.1.

3.2.4.2 Free Space Path Loss Calculation

In the link budget calculation, the operational frequency was assumed to be at 3 GHz. The loss caused by the distance between the satellite and a ground station was calculated using Equation 2.4.1.

Weather dependant attenuation will affect the link budget uniquely during each pass. The detailed effect on the system is not taken into consideration, save for noting the fact that at some points in time the link budget will be lower than this "blue sky-optimistic" calculation predicts.

3.2.5 SNR

SNR components, i.e., signal strength and noise strength, are computed in Sections 3.2.5.1 and 3.2.5.2.

3.2.5.1 Signal Strength

The signal strength is affected by both the transmitted signal power, the gain added by the antennas as well as in route losses. These terms are summed in Equation 3.2.5.

$$P_{SignalRX} = P_{SatTX} + G_{SatAntenna} + G_{GSAntenna} - L_{FS} \quad (3.2.5)$$

The signal power used in the link budget calculation is that transmitted by the satellite. The reason for this choice is that the satellite's transmit power is subject to the on-board power budget. The satellite's on-board power budget calculation falls outside of the scope of this project, but the imposed limit of 10 W is respected. With the transmit power known, all remaining factors of Equation 3.2.5 become functions of the Ground Station's orientation.

$$P_{SignalRX} = 40\ dB + G_{SatAntenna}(\alpha_{GS}) + G_{GSAntenna}(\alpha_{GS}) - L_{FS}(\alpha_{GS}) \quad (3.2.6)$$

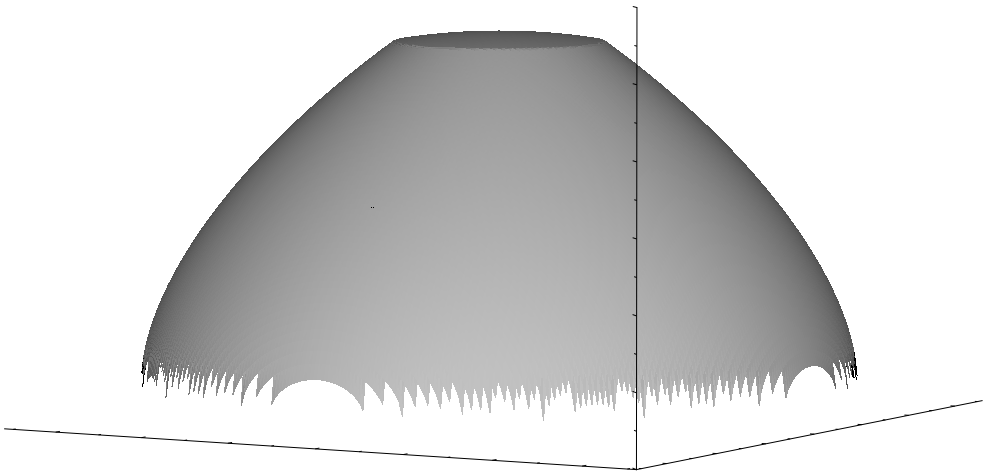


Figure 3.2.2: Signal Strength of a Pass (Without Ground Station Antenna)

A 3-D graphical representation of the signal strength, as calculated from Equation 3.2.6, is given in Figure 3.2.2. The graphical representation of Figure 3.2.2 is based on all possible elevation angles between GS and satellite.

3.2.5.2 Noise Floor

The noise floor is computed by obtaining the noise, which is allowed to pass through the band pass filter. The BW of the band pass filter is 25 kHz. At room temperature the amount of noise caused purely by cosmic background noise is given in Equation 3.2.7.

$$P_{NoiseFloor} = -174\ dBm + 10log_{10}(25k) = -130\ dBm \tag{3.2.7}$$

The result omits the effect of the noise added into the system by the transmitter, amplifier, antenna, receiver and cabling. This ideal noise floor is not a true reflection of actual noise and it is to be revised when the noise figures of the GS and satellite components are known. A more realistic estimate for $P_{NoiseFloor}$ was chosen at $-110\ dBm$ for the initial calculation phase of this project.

3.2.5.3 SNR calculation

The resulting signal power is compared to the noise floor, giving the SNR. A script was written to calculate the SNR across the horizon, as seen by a Ground Station. The results of the preliminary SNR calculations are presented in Table A.1 in the Appendix.

3.2.6 BER

The BER is of importance, since from that the FER is determined. From the FER, the effective throughput on the channel can be found. The BER for additive white Gaussian noise can be determined for the demodulation scheme QPSK per Equation 3.2.8 (Section 2.4.4).

$$BER = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{3.2.8}$$

3.3 Optimal Segment Length - Model 1

In Section 2.7.1.2 the satellite network stack was presented. The problem researched in this section, is to determine the segmentation length of the segments at the SS. The metric of measurement is the effective goodput or effective data throughput, which is a baud rate agnostic measure.

In order to establish the optimal frame length of a segment, it is important to find an expression for the effective data throughput. Effective data throughput is dependant on protocol overhead per frame, error correction coding efficiency and the channel BER.

3.3.1 Protocol Overhead

The protocol overhead per frame is that added by the SS header (1 *Byte*), the TS header (5 *Bytes*), error control field (2 *Bytes*), the start sequence (2 *Bytes*) and tail sequence (8 *Bytes*) of the SCCS. This overhead amounts to 18 *Bytes* per segment.

3.3.2 Coding Efficiency

The error correction coding of the SCCS makes use of $(63, 56)$ BCH code blocks. To the BCH code block a 1 *bit* filler is added. This means that the data is transmitted at an efficiency of 56 *bits* data for every 64 *bit* code block or $\frac{7}{8}$. If the data is less than 56 *bits*, then filler bits are introduced.

3.3.3 Frame Loss

If, at the TS, a frame is found to contain an error, the frame is discarded. In the event of a discarded frame the payload data is lost and the frame needs to be resent. A frame will contain an error if any of the following are true:

- Any of the code blocks of that frame contains a residual error after correction (i.e. there were two or more errors in that code block.)
- The start sequence has been missed (i.e. there were more than two errors in the start sequence.)

- The stop sequence of the previous frame was incorrectly accepted (i.e. a specific combination of two or more errors occurred.)

The expressions for these error probabilities may be determined using combination calculations, but have been taken from Annex D of [25]. The expressions of each of the error probabilities are given in the following text.

3.3.3.1 Error in Code Block

The BCH code blocks used in the SCCS of the TC-SDLP is a single error correction (63,57) code. The number of bits comprising a code word has been reduced to 56 bits. This reduction allows for some design improvements.

The first of these design improvements is that a 56 bit code word implies a natural number of bytes (7 bytes). Working with a natural number of bytes is desirable in a byte-orientated computing environment. Furthermore, the FEC and ECC bits occupy the last byte in the (64,56) code word.

The second advantage is the ability to add a parity check bit. The parity check allows for error detection in code words, where more errors occurred, than can be corrected. Hence, any single error can be corrected and, due to the parity check, every double error can be detected.

The probability of any code block containing residual errors, after decoding, is that of there being two or more errors in the code word. Where p is the BER and N is the number of code words, $N = \lceil \frac{L+8}{7} \rceil$, resulting from a segment length with L *Bytes*, then the probability of at least one residual error in the code words is given in Equation 3.3.1.

$$P_{CY}(p, N) = 1 - [(1-p)^{63} + 63p(1-p)^{62}]^N \tag{3.3.1}$$

3.3.3.2 Missed Start Sequence

If the start sequence preceding the encoded data is missed, the SCCS will remain in the "Searching State" and the encoded data is at least partially missed. The start sequence is accepted, even with one error present. Hence the probability of missing a start sequence, is that of two or more errors occuring in the start sequence. The probability of missing the start sequence, where p is the BER, is given in Equation 3.3.2.

$$P_{SY} = 1 - [(1-p)^{16} + 16p(1-p)^{15}] \tag{3.3.2}$$

3.3.3.3 Missed Tail Sequence

If a tail sequence is missed (falsely accepted) then the start sequence of the following code block may be missed. Additional data appended to the frame is disposed of at the TS, hence the preceding frame is not adversely affected by an incorrect acceptance of a tail sequence. The following start sequence

may however be missed due to the fact that the SCCS may not have gone into "Searching State". The probability of falsely accepting and decoding a tail sequence, where p is the BER, is given in Equation 3.3.3.

$$P_{TY} = 1953p^2(1-p)^{61} + 651p^3(1-p)^{60} \tag{3.3.3}$$

3.3.3.4 Frame Rejection due to Errors

The probability of having to reject a frame due to residual errors is the combined probability of Equation 3.3.1, 3.3.2 and 3.3.3. These equations have been summarised in Equation 3.3.4, which is a function of the BER and the length of the segment L.

$$FER(BER, L) = P_{TY} + (1 - P_{TY})(P_{SY} + (1 - P_{SY})P_{CY}) \tag{3.3.4}$$

3.3.4 Effective Data Throughput in terms of BER

Section 3.3 has shown that the effective data throughput can be characterised in terms of the physical layer quality BER and the length of the segments. The combined effective throughput model is given in Equation 3.3.5.

$$eff(BER, L) = (1 - FER(BER, L))(\frac{L}{8\lceil\frac{L+8}{7}\rceil + 10}) \tag{3.3.5}$$

Figure 3.3.1 displays the curve resulting from Equation 3.3.5. The BER has a range from 0 to 0.009 and the length is limited to 1016 bytes. The three axes are frame length and BER on the x- and y-axis respectively and effective throughput on the z-axis. The colour gradient of the graph in Figure 3.3.1 gives an indication of the throughput in various regions.

3.3.5 Frame Length Maximising Throughput

Equation 3.3.5 of Section 3.3.4 quantified the effective throughput at a given BER and frame length combination. The question of which frame length will maximise the effective throughput for a given BER needs to be asked. Answering this question will enable a policy decision to be made about the frame length for a given BER. The curve $L = f(BER)$, which maximises the effective data throughput, will infer the solution to the question. Sections 3.3.5.1 and 3.3.5.2 investigate the relationship $L = f(BER)$.

3.3.5.1 Differentiation

Equation 3.3.5 can be partially differentiated in terms of L. Solving for the length then returns the expression $L = f(BER)$.

Though the underlying principles of this method are sound, the method was abandoned for the following reasons:

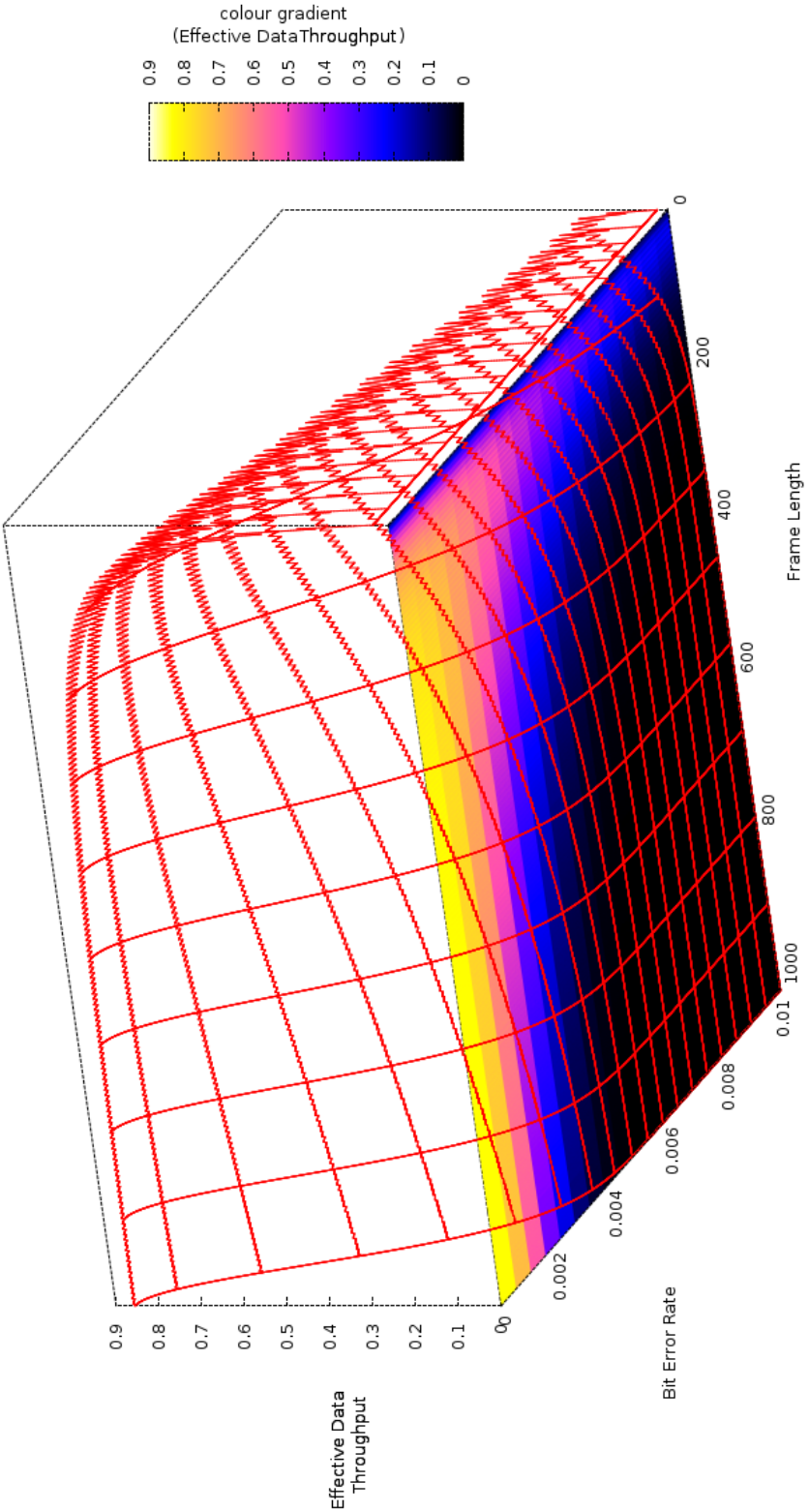


Figure 3.3.1: Effective Data Throughput for Various BER and Frame Length Combinations

- The ceiling operator, $\lceil x \rceil$, is 0 where $x \notin \mathbb{Z}$, and undefined where $x \in \mathbb{Z}$. This property of the ceiling operator implies that all ceiling operators need to be removed before differentiation, leading to inaccuracies.
- The chain rule of differentiation results in both a FER and a FER'. Both FER and FER' have elements that have powers of L and BER.
- Simplifying the resulting equation to a usable form is not possible.

3.3.5.2 Inspection

Since finding $L = f(BER)$ by differentiation is too cumbersome, the equation is found by inspection. This is a brute force method in which numerous values of BER are sequentially inspected. For each value of BER, the discrete values of the frame length, L, are cycled through, inserting all possible BER - L combinations into Equation 3.3.5. The frame length L, which maximises the effective throughput for a given BER, is stored. These BER frame length pairs are then used to construct the equation $L = f(BER)$.

3.3.6 Summary of Model 1

The model developed in Section 3.3 is a tool which can aid in the choice of an optimal segment length if the SNR or BER is known or remains constant. Using this model to adapt the segmentation length strategy during a communications window is only possible if the physical link quality is actively measured.

3.4 Optimal Segment Length - Model 2

The model of Section 3.3 provided a method of determining the optimal segmentation length if the BER is known. Since measuring the link quality is an impractical task, calculations done prior to the current communications window would have to be used to determine the optimal segmentation length strategy for the current communications window.

The strategy presented in Section 3.4 aims to attain a metric, already common to the datalink layer, to determine the link quality. This metric is the FER. The FER will be used to determine the optimal segmentation length during each pass.

3.4.1 Effective Data Throughput in terms of FER

In Section 3.3.4, Equation 3.3.5 was presented, which determined the effective data throughput based on the BER and frame length. Since frames are acknowledged in the TC-SDLP, the metric FER is known at this layer. By

simplifying FER to be a new variable of Equation 3.3.5, which is independent of length and embodies BER, an optimal segment length model can be devised using only FER and segment length. Equation 3.3.5 has been simplified to Equation 3.4.1:

$$eff(FER, L) = (1 - FER)(\frac{L}{8\lceil \frac{L+8}{7} \rceil + 10}) \tag{3.4.1}$$

Next, the method of inspection described in Section 3.3.5.2 is applied to find the optimal segment length in terms of the FER. Figure 3.4.1 shows the optimal segment length vs FER. One point worth noting is that Figure 3.4.1 does not show a graph. This is because the FER is a function of the segment length and not vice versa. Nonetheless, the FER is the measurable quantity by which the optimal segment length is approximated. Equation 3.4.2 gives an approximation of the curve in Figure 3.4.1.

$$L_{opt} = \lceil \frac{\lceil 18 - 16FER \rceil}{7FER} 7 - 1 \rceil \tag{3.4.2}$$

3.4.2 Frame Error Rate Estimator

In Section 3.4.1 it has been shown that, if the FER is known exactly, the optimal segment length can be determined. Since determining the exact FER is a statistical process, a best effort estimate has to be made. This estimate of the FER is detonated as FER estimation (FÊR).

The method in which the FÊR is estimated and the information used to make this estimation are described in the following sections.

3.4.3 Choosing an FÊR estimation scheme

As mentioned in the Section 3.4.2, the exact FER can only be estimated. The work in Section 3.4.3 describes three estimation methods which were investigated and compared. The results of this endeavour are given and the most appropriate method was used in the implementation.

An initial approach used to obtain the FÊR, is to divide the total number of failed frames by the total number of transmissions. Such an initial approach can easily be implemented and would give the true average FÊR. The disadvantage is that this FÊR will hide the variation of the real FER, which varies with time over the communications link.

The fact subsequently emerges that a shorter history is desired to obtain the FÊR. An analogy to this problem is the use of a capacitor on a noisy, information-bearing line. Using a capacitor that is too big, will result in the message signal being equal to the average voltage of the message. Using too small a capacitor will allow too much noise onto the information signal.

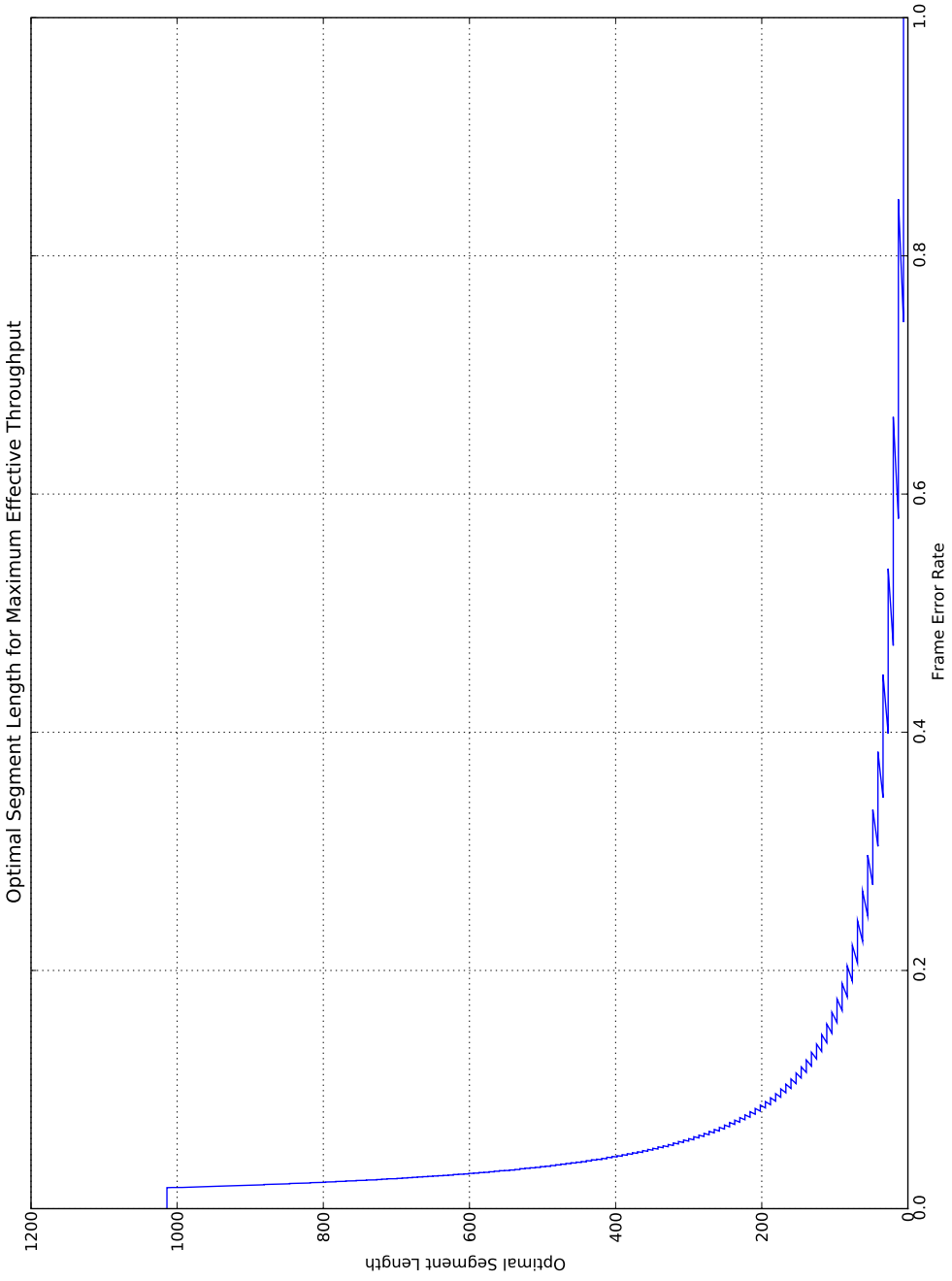


Figure 3.4.1: Optimal Frame Length vs FER

Sections 3.4.3.1, 3.4.3.3, 3.4.3.4 and 3.4.3.5 investigate the distribution of the FÊR and the estimation schemes used. The scenarios to which the simulations were subjected are discussed in Section 3.4.4. The results obtained from the simulation runs are presented in Section 3.4.4.2. Finally, the choice made in terms of the FÊR scheme is presented.

3.4.3.1 Distribution of FÊR

The FÊR will be obtained from a predetermined number of history frames. The first step in the process of finding the best estimation scheme is to investigate the distribution of the FÊR.

The history has a fixed length, this quantity being denoted by 'H'. The probability of any frame in H being corrupted is FER. The FÊR is obtained by dividing the number of lost frames in H (denoted by 'l') by H. Since this problem is a choose l from H, the distribution of the FÊR will be Binomial.

The Binomial distribution is a discrete distribution, described in general by the formula shown in Equation 3.4.3.

$$f_x(x) = \sum_{k=o}^{N} \binom{N}{k} p^k (1-p)^{N-k} \delta(x-k) \tag{3.4.3}$$

with

$$\overline{X} = Np$$

$$\sigma_X^2 = Np(1-p)$$

Equation 3.4.3 gives a general formula for impulses at $x \in \{0; N\} \cap \mathbb{Z}$. This distribution is normalised to the range $x \in \{0; 1\}$ by multiplying x by H, the history length. Equation 3.4.4 gives the normalised formula as applicable to the FER.

$$f_x(x) = \sum_{k=o}^{H} \binom{H}{k} FER^k (1-FER)^{H-k} \delta(Hx-k) \tag{3.4.4}$$

with

$$\overline{X} = FER$$

$$\sigma_X^2 = \frac{FER(1-FER)}{H}$$

3.4.3.2 Improved Estimation Schemes

After the binomially distributed FÊR is obtained, the next step in the process is to find the best next FÊR. Three methods of obtaining this estimate which were investigated are:

1. The next FER is exactly the same as the current. Hence $F\hat{E}R_{N+1} = F\hat{E}R_N$
2. Fitting a curve, using the least squares method, to the past data, to obtain the next point. Hence $F\hat{E}R_{N+1} = m(N+1) + c$.
3. Using filtering to obtain the best estimate. The Kalman Filter was used in this regard.

These three methods are investigated in Sections 3.4.3.3, 3.4.3.4 and 3.4.3.5.

3.4.3.3 Static FÊR

The static FÊR approach suggests the best guess for a next FÊR the current FÊR. The procedure used for this approach starts out by counting the lost frames, 'l', in H and dividing them by H. This is then the FÊR for the current point in time, which is used as a best guess for the next FÊR.

3.4.3.4 Least Squares

In regression analysis the method of least squares is used to fit an nth order polynomial to a set of data which is greater than the number of unknowns in that set. The concept of the method is to find the difference between the estimated polynomial and the reading. This difference is the error, or residual, which is squared and summed. The smallest sum will be the best fit.

In the work described in this section a first order polynomial, a line, is fitted to the data. The slope, m, and the y-intercept, c, of the curve are found by the well-known formulae:

$$m = \frac{h(\sum xy) - (\sum y)(\sum x)}{h(\sum x^2) - (\sum x)^2} \tag{3.4.5}$$

and

$$c = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{h(\sum x^2) - (\sum x)^2} \tag{3.4.6}$$

Once the fitted curve is found, the next reading is taken from this curve. This reading will be at time instance $N+1$. It is hoped that the reading at $N+1$ will be a better estimate than that of a static prediction.

There are two points of concern which should be noted about using this approach. Firstly, the least squares approach relies on the data being normally distributed, which it is not (see Section 3.4.3.1). Secondly, this approach

is not usually used for prediction because of the fact that a fitted curve may cross the bounds of sane readings.

In [40], fancy footwork was done to justify the use of the least squares method on binary data. Although the results may be biased at small and large FER, the estimation method is worth investigating, due to the simplicity of Equations 3.4.5 and 3.4.6.

The fact that a next prediction may not give a sane result, such as a negative FÊR or a FÊR > 1, must be noted, but a simple work around will correct these prediction errors. If the predicted value is greater than 1, then then it is set to 1, else if the prediction is less than 0, then the prediction is set to 0, hence maintaining sanity.

3.4.3.5 Kalman Filter

The Kalman filter, named after the Hungarian mathematician, Rudolf E. Kalman, is a recursive filter which obtains a signal from a set of noisy measurements. The filter receives a new reading and uses it to correct the current estimation. After correcting, a time update is made and the next prediction is made. The correction and prediction steps are repeated for each time step.

Figure 3.4.2, taken from [41], shows each of the equations pertaining to the filtering process. The order of updating is clockwise. When all the equations have been updated the new estimate of the state x, $\hat{x}_k^-$, is obtained.

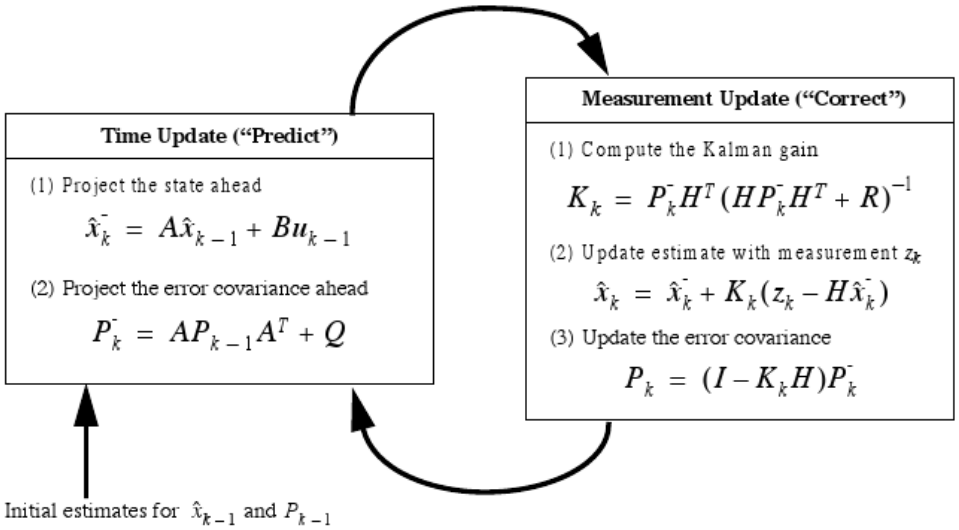


Figure 3.4.2: Kalman Filter Update and Correction procedure

These equations can be simplified significantly when applied to the prediction of a single variable, of which the current state is measured directly.

$$p_k^- = p_{k-1} + q \tag{3.4.7}$$

$$\hat{x}_k^- = \hat{x}_{k-1} + K_k(z_k - \hat{x}_k^-) \tag{3.4.8}$$

$$p_k = (1 - K_k)p_k^- \tag{3.4.9}$$

$$K_k = \frac{p_k^-}{p_k^- + \frac{\hat{x}(1-\hat{x})}{h}} \tag{3.4.10}$$

The variable z is the the measurement of the current state of FÊR. The fraction $\frac{\hat{x}(1-\hat{x})}{h}$ is the variance of the normalised binomial distribution. Now that the three methods of estimation have been described, focus can be shifted to evaluating them.

3.4.3.6 Computational Complexity

In a simulation environment the computational complexity of these methods is not of much concern, but when it comes to the implementation, computation becomes important.

A lost frame has a binary state, it is either lost '1' or not '0'. The history of lost frames can therefore be maintained in a bit array. A new event can be introduced by a bitwise shift. Counting the lost frames is as simple as counting the high bits in the history. Obtaining the FÊR would divide lost frames by history length, a known quantity.

The estimation of the FER by the static FER method would only involve obtaining the current FÊR. Obtaining the current FÊR has been shown to be trivial.

The method of least squares would require the knowledge of a number of previous FÊRs and their places in time, in other words their x and y coordinates. A First In First Out Queue (FIFO) can be used for this purpose.

Besides the queues that maintain the history and the past FÊR, Equations 3.4.5 and 3.4.6 have to be solved. Then the predicted value of FÊR can obtained by evaluating the point at the next time increment.

The Kalman filter would require the updating of four variables by evaluating Equations 3.4.7 through 3.4.10.

Three methods, ranked by complexity, are: Static FER, Kalman Filter and, the most complex, the Least Squares method. The computational complexity may become a deciding factor in the case of marginal improvement in FÊR.

3.4.4 Simulation and Scenarios

The three estimation methods have been explained and in this section their performance is compared. The performance is quantified by means of simulation. The estimation methods are subjected to various scenarios.

The first question to be answered is how the performance of an estimation method can best be measured. Referring back to Figure 3.4.1, it may be seen that a small change in FER in the regions FER ≤ 0.2 will require a large change in frame length. Whereas a large change in the region FER > 0.2 will require only a small change in frame length. This means that a greater sensitivity is desired in the region FER ≤ 0.2. Hence the estimation method that makes the smallest error in the specified region is the best choice.

When creating a simulation, the following question to consider is which aspects of the environment can be excluded, and which need to be included in the model. The goal of this simulation is to determine which estimation method best follows the FER, therefore the fact that the FER is a function of BER is ignored. Frames have a length and therefore the time they take varies according to this. This fact was also ignored.

The simulation was written to have N discrete time steps. The FER was predetermined by the scenario that the simulation was subjected to. At each time step a random number determines whether the frame was lost and/or corrupted. The three estimation methods are used to obtain a $\hat{\text{FER}}$ for each scenario, at the discrete time steps.

3.4.4.1 Scenario Selection

Since the change of BER due to weather attenuation, cannot be predetermined, the corresponding change in FER is indeterminable. Testing the various models was done using fabricated data. The use of varying scenarios will help expose tendencies and determine the conditions under which each of the respective estimation methods performs better.

The scenarios used involved:

- Constant FER steps (Block wave)
- Linear increasing FER (Sawtooth)
- Exponentially increasing FER
- Sinusoidal FER (Chirp)

The three estimation methods were combined with various history lengths. The combination of estimation methods and history length gave a grid. The location in the grid where the values for the mean and standard deviation are minimised, is the combination of estimation method and history length which leads to the best $\hat{\text{FER}}$.

3.4.4.2 $\hat{\text{FER}}$ Simulation Results

The scenarios were run over one million samples and the results showed an acceptable amount of deviation. The resulting mean and standard deviations

are given in Table 3.4.1. Table 3.4.1 was the first iteration of the selection process, hence history lengths of 50, 100, 200 and 400 were selected. From Table 3.4.1 the area to be focused on was identified.

The method of least squares was outperformed by the competing Static FÊR and Kalman Filter methods. The history length of 100 performed best for both the Static FÊR as well as the Kalman Filter approach. The process of selecting a best estimation method and its corresponding history length was then repeated, this time using multiples of 8 around the history length close to 100. The rationale for using multiples of eight is that this corresponds with using bytes to store the history of lost frames. The Least Squares method was excluded from the refined search. Figure 3.4.3 shows an exhibit of the various estimation methods under the sawtooth scenario and Figure 3.4.4 shows the histogram of the full run.

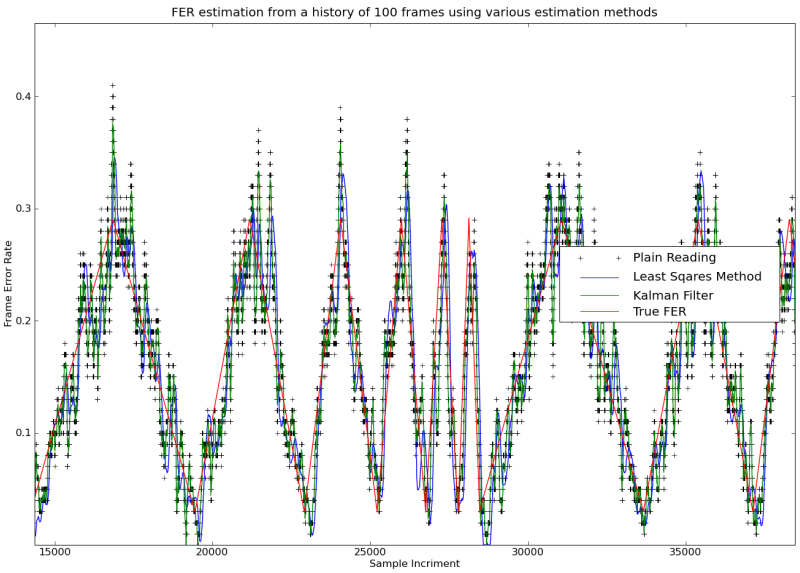


Figure 3.4.3: Extract of FER and FÊR

Table 3.4.2 gives the results obtained from the refined search. The history sizes ranged from 88 through 112. At this point it must be noted that if the Static FÊR method performs similarly to the Kalman Filter, its simplicity makes it the better choice. Nonetheless, the search has revealed that the methods performed very similarly and the best estimation method is that of Static FÊR and a history length of 96. This endeavour has shown that the best

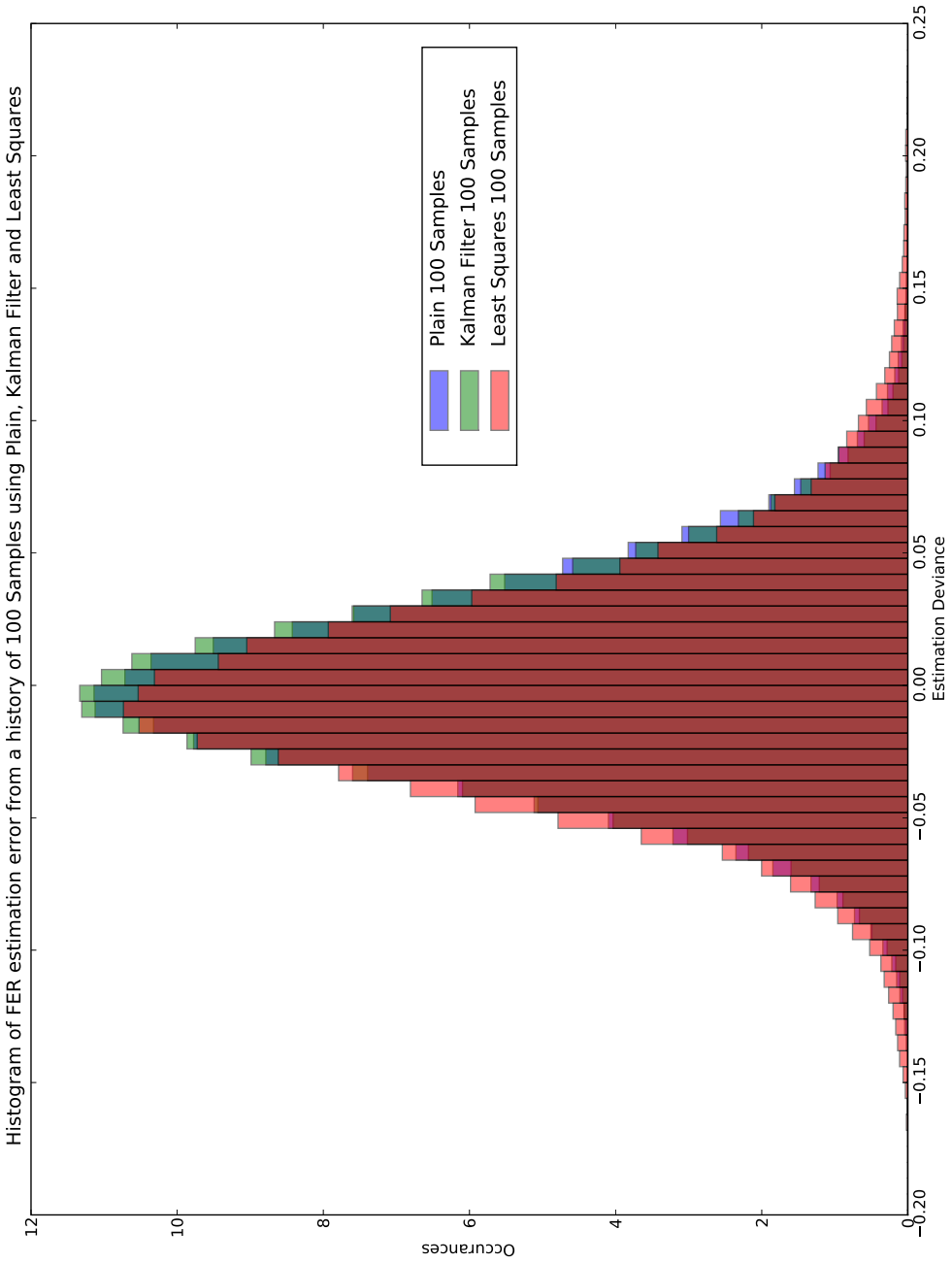


Figure 3.4.4: Histogram of (F̂ER - FER)

Table 3.4.1: Comparison of FÊR methods over History Sizes ranging from 50 to 400

		Block Wave		Sawtooth		Exponential		Chirp		Average	
		Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Static FÊR	50	-0.000420	0.052184	-0.000156	0.056613	-0.000502	0.050886	0.000017	0.054672	-0.000265	0.053589
	100	0.000368	0.039996	-0.000235	0.051395	0.000081	0.038331	0.000014	0.052701	0.000057	0.045606
	200	-0.000536	0.034926	0.000129	0.062339	-0.000821	0.035601	-0.000403	0.066326	-0.000408	0.049798
	400	0.000267	0.038542	-0.000245	0.077368	0.000606	0.047356	0.000158	0.072652	0.000197	0.058980
Least Squares	50	-0.002381	0.041847	-0.002355	0.074061	-0.002702	0.040517	-0.002189	0.086113	-0.002407	0.060634
	100	-0.001673	0.042660	-0.002632	0.078978	-0.002170	0.042564	-0.002269	0.094452	-0.002186	0.064664
	200	-0.002723	0.043165	-0.002474	0.086730	-0.003118	0.047195	-0.002800	0.097790	-0.002779	0.068720
	400	-0.002171	0.044317	-0.002888	0.093262	-0.001808	0.057142	-0.002270	0.085156	-0.002284	0.069969
Kalman Filter	50	-0.000986	0.044673	-0.000877	0.053341	-0.001077	0.042921	-0.000758	0.052072	-0.000925	0.048252
	100	0.000187	0.038810	-0.000529	0.055805	-0.000104	0.036949	-0.000324	0.058168	-0.000192	0.047433
	200	-0.000591	0.036066	0.000012	0.067522	-0.000883	0.037388	-0.000537	0.071803	-0.000500	0.053195
	400	0.000248	0.040136	-0.000287	0.081063	0.000581	0.049784	0.000129	0.074778	0.000168	0.061440

Table 3.4.2: Comparison of FÊR methods over History Sizes ranging from 80 to 112

		Block Wave		Sawtooth		Exponential		Chirp		Average	
		Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Static FÊR	80	0.000459	0.042932	-0.000305	0.050869	-0.000052	0.041644	0.000025	0.050715	0.000032	0.046540
	88	0.000374	0.041391	0.000037	0.051002	-0.000270	0.040129	-0.000141	0.051511	0.000000	0.046008
	96	-0.000019	0.040597	-0.000293	0.051066	-0.000357	0.038444	0.000170	0.051427	-0.000125	0.045384
	104	0.000307	0.039377	0.000196	0.052367	0.000121	0.037436	0.000346	0.052826	0.000242	0.045501
	112	0.000189	0.038905	-0.000109	0.052128	0.000241	0.037009	-0.000022	0.053979	0.000075	0.045505
Kalman Filter	80	0.000193	0.040194	-0.000308	0.054715	-0.000497	0.037930	-0.000529	0.056138	-0.000285	0.047244
	88	0.000149	0.039354	-0.000604	0.055276	-0.000553	0.036795	-0.000183	0.056608	-0.000298	0.047008
	96	-0.000213	0.039161	-0.000088	0.057123	-0.000048	0.036295	0.000026	0.058547	-0.000081	0.047781
	104	0.000139	0.038434	-0.000364	0.057181	0.000089	0.036405	-0.000318	0.059969	-0.000114	0.047997
	112	0.000042	0.038299	-0.000364	0.057181	0.000089	0.036405	-0.000318	0.059969	-0.000138	0.047963

estimation of the current FER, if the FER varies over time, is obtained by the average of 96 history readings.

3.4.5 Summary of Model 2

Model 2, used for determining optimal length, uses a metric available to the datalink layer, the FER. Since the FER is a statistical process, the true FER is approximated as FÊR. Three estimation methods were compared and the simple average proved to be the most usable.

This model may still be criticised on one point, i.e. the information used to obtain the FÊR is based on the number of lost frames. The TS is responsible for the reliability and acknowledgement of frames. Segmentation, on the other hand, is performed in the SS. Using TS information to make SS decisions violates the independence of the TC-SDLP sublayers.

3.5 Optimal Segment Length - Model 3

Model 2, Section 3.4, discussed a method of determining the optimal segmentation length of a frame according to the metric FER. The metric FER is within the subset of information available to the datalink layer, but not to the SS.

This section discusses a final revision of the optimal segmentation length problem. A metric using only data available to the SS is now created to determine the optimal segmentation length. The new metric used is the time.

3.5.1 Information Available to SS

In creating a metric to determine the optimal segmentation length the information available to the SS is investigated. The SS has access to the following information:

- Whether the buffer, containing unacknowledged frames of the sliding window used by the TS, is full. (This is reasonable, because the SS should not pass data to the TS unless there is space available.)
- The system baud rate. (Remains static)
- The size of the segments passed to the TS. (The frames are segmented by the SS)
- The system time. (Time is common to the system and hence all programs should have access to this)

3.5.2 Determining Lost Frames

The SS may not query the TS as to whether a frame has been lost. This information is exclusively available to the TS. Notwithstanding, an attempt is made to approximate the status of the frames within the SS. If a successful approximation of number of lost frames can be made, then the FÊR can be obtained and the optimal segment length may be determined based on this.

The attempt to estimate whether a frame was lost is based on the fact that the SS would have a large amount of payload data, which is continually being presented to the TS. The TS can only accept new frames until the sliding window has been filled. Once the sliding window is full, the TS only accepts new frames from the SS if the oldest frame has been acknowledged.

Let the time taken between the SS presenting a frame to the TS and the TS accepting this frame be, Δt. If the oldest frame in the TS has been transmitted successfully, then the upper bound of Δt is expressed Equation 3.5.1 but, due to the sliding window mechanism, Δt is usually described by Equation 3.5.2. If the frame was lost or corrupted then Δt will be larger than the expected delay of Equation 3.5.1.

$$\Delta t_{norm\,MAX} = \frac{frame\,size}{baud\,rate} + \frac{CLCW\,size}{baud\,rate} + 2(RF\,Prop.) + (Processing) \quad (3.5.1)$$

$$\Delta t_{norm\,MIN} = \frac{frame\,size}{baud\,rate} \quad (3.5.2)$$

There are two cases in which retransmissions occur. The first of these cases is due to a retransmit flag being set in a CLCW. The second is due to a timeout occurring.

3.5.2.1 Lost Frames Indicated by Retransmission Flag

In the case of a retransmission flag being set, TS resends the frame. This retransmission will take the maximum time to transmit (Equation 3.5.1). Hence the frame retransmission due to a flag will take longer than the maximum transmission time (Equation 3.5.3).

$$\Delta t_{rmit\,flag} > \Delta t_{norm\,MAX} \quad (3.5.3)$$

3.5.2.2 Lost Frames Indicated by Timer

In the case of a timeout triggering the retransmission of a frame, the frame will, at most, have a delay of the timeout time plus the maximum delay (Equation 3.5.4). This is the maximum delay time for a single retransmission.

$$\Delta t_{rmit\,timeout} \leq (timeout\,time) + \Delta t_{norm\,MAX} \quad (3.5.4)$$

3.5.2.3 Single Retransmission

If a single retransmission occured, the delay, Δt, will be in the range indicated in Equation 3.5.5.

$$\Delta t \in (\Delta t_{rmit\ flag}, \Delta t_{rmit\ timeout}] \tag{3.5.5}$$

3.5.2.4 Multiple Retransmissions

In the unlikely event of a frame being retransmitted more than once, the number of retransmissions will fall within a natural number of $\Delta t_{rmit\ timeout}$. This means that if time falls within the interval expressed by Equation 3.5.6, then approximately n retransmissions occurred.

$$(n-1) \cdot \Delta t_{rmit\ timeout} < \Delta \leq n \cdot \Delta t_{rmit\ timeout} \text{ where } n \in \mathbb{N} - \{1\} \tag{3.5.6}$$

3.5.3 Summary of Model 3

Model 3, discussed in Section 3.5, involved using data available to the SS to make an estimate of whether a frame had failed. This information can then be used in the existing Model 2 to obtain an FÊR.

3.6 Simulation

A simulation is a process in which aspects of a given real world problem are modeled and placed under test conditions. The usefulness of a simulation depends strongly on simplifying the problem enough to make the implementation or coding effort less than implementing the real system, while keeping it accurate and valid. Hence the aim of the work described in this section is to build a valid, credible and appropriately detailed simulation model. The simulation work done in this section draws from the book [42].

Section 3.3 showed a definite increase in effective data throughput if the suggested optimal length is used. Even though the mathematical aspects of using the optimal segment length look promising, it was decided to test the optimal suggested segment length calculator in a simulation environment. The simulation was done to test and get a benchmark of the real system.

3.6.1 Conceptual Modelling

The first step in deciding on a simulation design is to define a conceptual model of the real world problem. This step involves specifying relevant as well as irrelevant aspects of the system. In the case of the optimal length estimation for the TC-SDLP protocol, it is important to decide which of the specifications will affect the correctness of the model and which will not.

3.6.1.1 Unit of Measure

The unit of measure is a quantity which measures the performance of a simulated scenario. The unit of measure should be simple and yet carry enough information. Since the problem discussed in this work deals with attempting to maximise the effective data throughput, the unit of measure was given as the fraction of total successfully transmitted information data over total transmitted data. Such a simple unit of measure makes comparison of scenarios a systems trivial task.

3.6.1.2 Key Specifications

The simulation should give a good estimate of how the real system will follow the optimal segmentation length. The simplifications made for the simulation are:

- There is an endless amount of data waiting to be serviced by the data link layer.
- The segmentation sublayer knows about every failed frame.
- Acknowledgements do not fail.
- FER is estimated from the real history (Model 2).

Aspects which were not simplified:

- The optimal segment length is obtained from the $\hat{\text{FER}}$.
- Frames fail based on BER, not FER.
- Failed frames are resent.

The variables which were used in the simulation:

- The history length used to estimate FER.
- The frame buffer size.
- BER.

3.6.2 Simulation Program

The simulation of the datalink layer was done by means of a simulation program. The University of Stellenbosch Computer Science department has developed a simulation framework in C++, which was used and extended for this project. The simulator classes may be found in Figure 3.6.1.

The segment class derived from the actor class represents a segment. Methods setting the required estimated segment length, as well as showing the

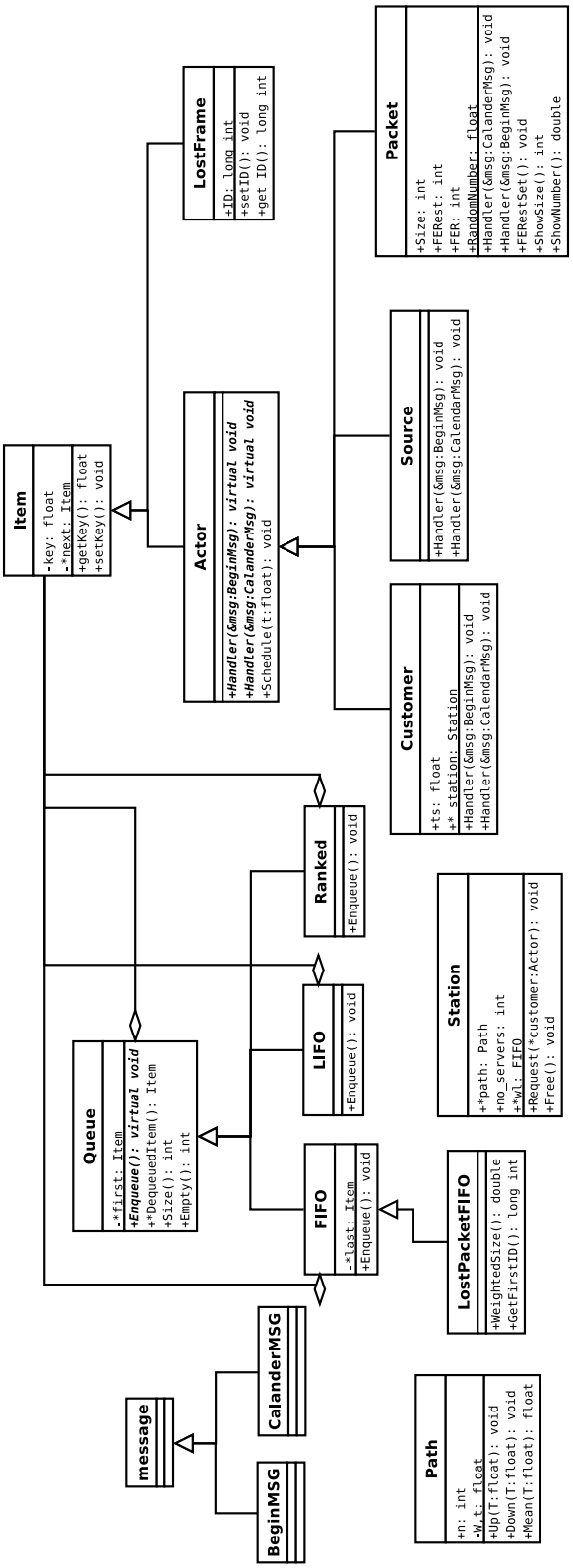


Figure 3.6.1: Simulator Classes UML

segment length, were added to this method. The segment length estimation method implements the approximated optimal segment length of Section 3.4. The lost frame class has been derived from the item class. It has one additional property ID and all associated methods have been created.

The FIFO class is used to represent the FOP-1 sliding window. This class gives all the functionality of a FIFO queue and this is sufficient to model FOP-1. The queue containing the lost packets is also a FIFO queue but, because of some additional required methods, a derived class was created. This class, lostframeFIFO, has the additional methods to obtain the ID of the first packet. The remaining classes have been used unaltered.

The functions created were FERest and FERtrue. FERest estimates the FER from the state of the lost frame FIFO and effectively obtains the value of $\hat{FER}$ as described in Section 3.4. The FERtrue calculates the true FER from the BER as well as the frame size at that specific instant.

3.6.2.1 Program Structure

A packet moving through the TC data link layer model is assigned one property at inception. This is the length, which is obtained from the optimal frame length calculator. Once this packet reaches the front of the queue the FER is calculated from the BER at that time, as well as the length of the packet. This value is compared to a random number between 0 and 1 inclusively, obtained from the gsl_rng library [43]. If the FER is smaller than the random number, then the frame has successfully been received, else the packet has failed.

If the packet is successfully received its value is added to the total payload data. The packet length is magnified by the redundancy, and the true amount of data transmitted is summed to the total data transmitted. The packet is dequeued and a new packet is created, assigned a length and placed onto the FOP-1 queue.

If the packet has failed then the packet length is magnified by the redundancy and this value is summed to the total data transmitted. A new lost frame is created and placed onto the frame history queue.

In both cases the number of events is incremented. The first lost frame in the frame history queue is dequeued if the difference between its ID field and the current ID is equal to the history length.

The above process is repeated until the number of events are as indicated by the user, signifying one simulation run. The result of such a run is the a percentage of effective data throughput. The run is repeated numerous times to assure good confidence in the results obtained. Each of these runs is repeated with a unique random number seed.

3.6.2.2 Initialisation

During the initialisation phase of the simulation, the user is prompted for: the history frame length, the weighting of the samples, the number of frames in FOP-1, the number of events to be simulated, the BER at the start of the simulation and the BER at the end of the simulation.

The frame history is initialised with an alternating sequence of successes and failures. Each lost frame is assigned an ID indicating its position in time. The FOP-1 queue is initialised with a number of packets, equal to the user defined FOP-1 length. These packets have a length obtained by the Optimal Frame the optimal length calculator.

3.6.2.3 Simulation Results

The simulation results are given alongside the results of the real system in Chapter 5. The results of the simulation give a benchmark to which the results of the real system may be compared.

3.7 Conclusion

Chapter 3 first investigated the link budget of the system. The link budget showed that the link quality varied greatly during each pass. The link budget did not take into account attenuation due to weather conditions. Furthermore, it was shown that adapting the segmentation length leads to an increase in throughput.

Three models for obtaining optimal segmentation length were presented. Model 1 gave a formula in which the frame length could be adapted to optimize throughput. This model used the link quality to determine the optimal segmentation strategy. Since the link quality is not known in the datalink layer the model was updated to Model 2.

Model 2 gave a relationship between optimal segmentation length and a metric known within the datalink layer, the FER. The FER cannot be known exactly, hence estimation methods were investigated to approximate the FER. The FÊR was introduced as an estimation of the FER and various methods of attaining such an estimate were compared.

Since the mechanism which notifies the datalink layer about lost frames is in the TS, and segmentation is performed in the SS, Model 2 was updated to Model 3. Model 3 formulated a method to estimate whether a frame was lost based purely on the information available to the SS, namely time. The static FÊR method, obtained from Model 2, was used to obtain the FÊR. From the FÊR the optimal segmentation length, which maximizes throughput, could be selected.

Next, attention was turned to simulation. The simulation was based on a simulation framework developed by the University of Stellenbosch. The

simulation model provides a benchmark to measure the real world performance.

The next chapter presents a discussion on how the TC-SDLP was implemented on the SuperH-4 (SH-4) microprocessor. The implementation of the TC-SDLP is then used to obtain results, which are presented alongside those of the simulation in Chapter 5.

Chapter 4

Implementation

The TC-SDLP was introduced in Section 2.6.3.3 and has been selected as the MCSP datalink layer protocol. This chapter presents a discussion of the implementation of the TC-SDLP standard [25]. The software implementation involved a substantial C coding effort.lines of C code.

4.1 Hardware and Software Platform

The prototype datalink protocol software was implemented for a specific hardware and software platform. The platform is the same as that used in Sumbandilasat. In the following Sections, 4.1.1 and 4.1.2, the hardware and software of the satellite are discussed.

4.1.1 SH-4

The SH-4 is a Reduced Instruction Set Computer (RISC) microprocessor. The SH-4 is a product of Hitachi and was introduced in 1998. The SH-4 was chosen for the CPU due to its ability to function correctly even after prolonged radiation exposure. Space equipment is exposed to large amounts of radiation and the SH-4 was tested by SunSpace and found to be adequately radiation insensitive.

4.1.2 QNX

The datalink layer was programmed for the Quick uNiX (QNX) Neutrino operating system. QNX Neutrino is very similar to the UNiplexed Information and Computing System (UNIX) operating system. The fact that QNX Neutrino is a Real Time Operating System (RTOS), that it conforms to Portable Operating System Interface (POSIX) and that it employs a microkernel set the QNX Neutrino apart from UNIX.

A RTOS assures the meeting of time based software constrained. In the literature RTOS has been defined as follows: "A realtime system is one in

which the correctness of the computations not only depends upon the logical correctness of the computation but also upon the time at which the result is produced." [44] The use of RTOS has become popular in the embedded space environment.

The POSIX standards, defined in the IEEE Std 1003.x documents, specify interfaces for software and the manner in which these should be used by system and users. Conformance to these standards offers software greater portability.

4.2 Datalink Program Overview

The program interface conforms to the POSIX interface standard. The datalink layer offers a standard interface in the "/dev" file system. Each channel is identified by a unique identifier "/dev/ch*N*" where *N* is the channel number.

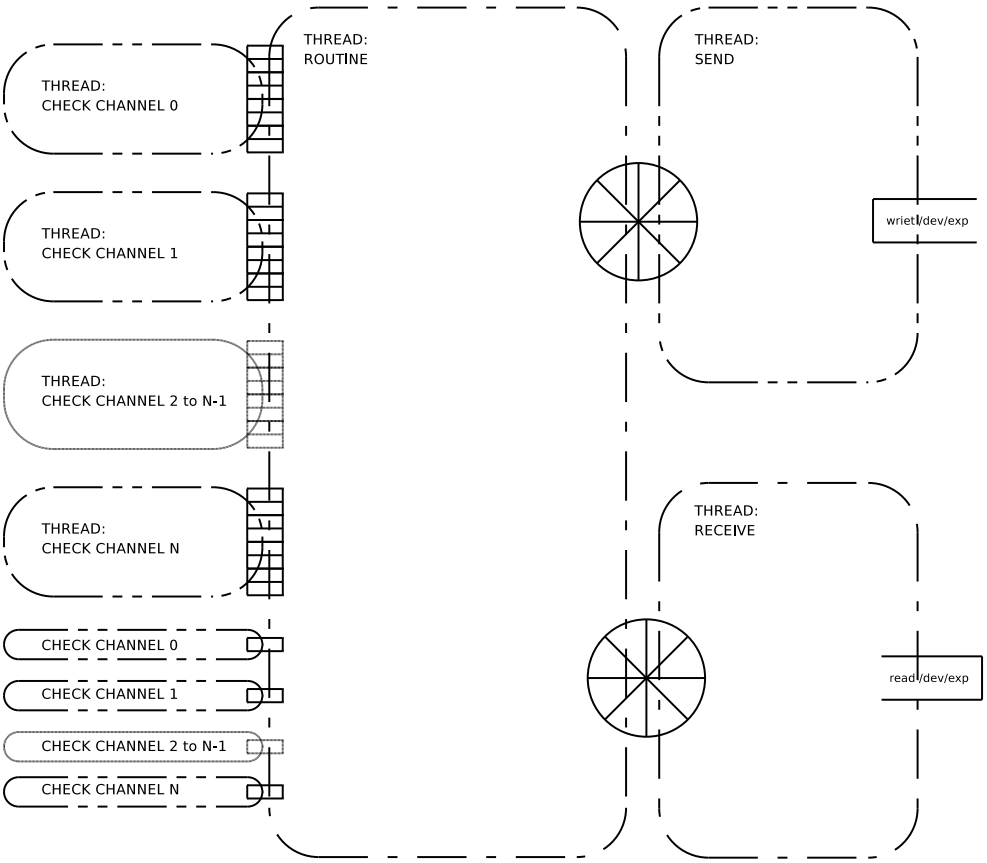


Figure 4.2.1: Datalink Program Overview

The datalink program is structured in a modular fashion. Although all three sublayers are contained within a single program, each sublayer has access only to data which is a part of its information subset.

The main program contains several aspects which need to occur concurrently. Writing to the physical layer should not be affected by the reading process. Likewise each channel being accessed by the higher layer should not influence the performance of the other channels.

Hence the processes of writing to and reading from the physical layer, of the SCCS, are each handled by a different thread. The TS process, the main routine, is contained within a thread. Finally each channel of the SS needs to run concurrently with the others and each is therefore handled by a separate thread.

The conceptual structure of the programing logics is depicted in Figure 4.2.1. Each thread is depicted by a rounded rectangle with broken lines. The detailed flow diagrams of the threads are discussed in the Sections 4.3, 4.4 and 4.5. The following sections, 4.2.1 and 4.2.2, discuss the overview of the program logic as well as the data structures used.

4.2.1 Datalink Program Flow

The main program is a straightforward process starting various threads. The flow diagram is depicted in Figure 4.2.2.

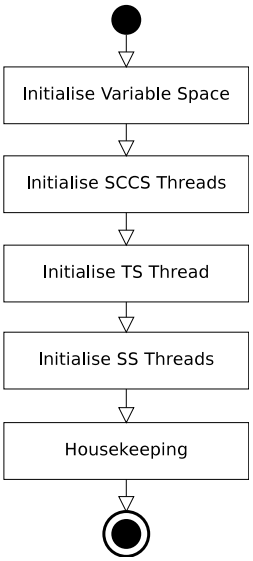


Figure 4.2.2: Datalink Program Flow Overview

When the program is started, the threads are initiated in the following sequence:

- First the SCCS threads, the send and receive threads, are spawned. The details of these threads are given in Section 4.3.4 and 4.3.5.
- Next the TS thread, the routine thread, is spawned. The details of this thread are given in Section 4.4.5.
- Finally the SS threads, the check channel threads, are spawned. The details of these threads are given in Section 4.5.4.

4.2.2 Datalink Data Structures

Data are passed between the SS and the TS by means of a data structure named "Sliding Window Data Structure". Data are passed between the TS main routine and the SCCS transmit and receive threads by means of a data structure named "Buffer Data Structure".

4.2.2.1 Sliding Window Data Structure

The Sliding Window Data Structure is the structure that is used to pass frames between the SS receiving thread and the TS routine thread. Hence it is accessed by two different threads. This requires a thread safe design for the Sliding Window Data Structure.

The Sliding Window Data Structure contains frames and needs space for the "Positive Window" number of frames. The design of the "Positive Window" length is elaborated on in Section 4.4.2.3. The variables maintained in a Sliding Window Data Structure are:

- Back - Indicates the oldest unacknowledged frame
- Next - Indicates the next frame to be transmitted
- Front - Indicates the position where the next frame should be placed
- Back Lock - A mutex that protects the "Back" variable from multiple access
- Front Lock - A mutex that protects the "Front" variable from multiple access
- Time Stamp - The time at which the front frame was either last transmitted or a frame was last acknowledged
- Counter - The amount of times the front frame was transmitted

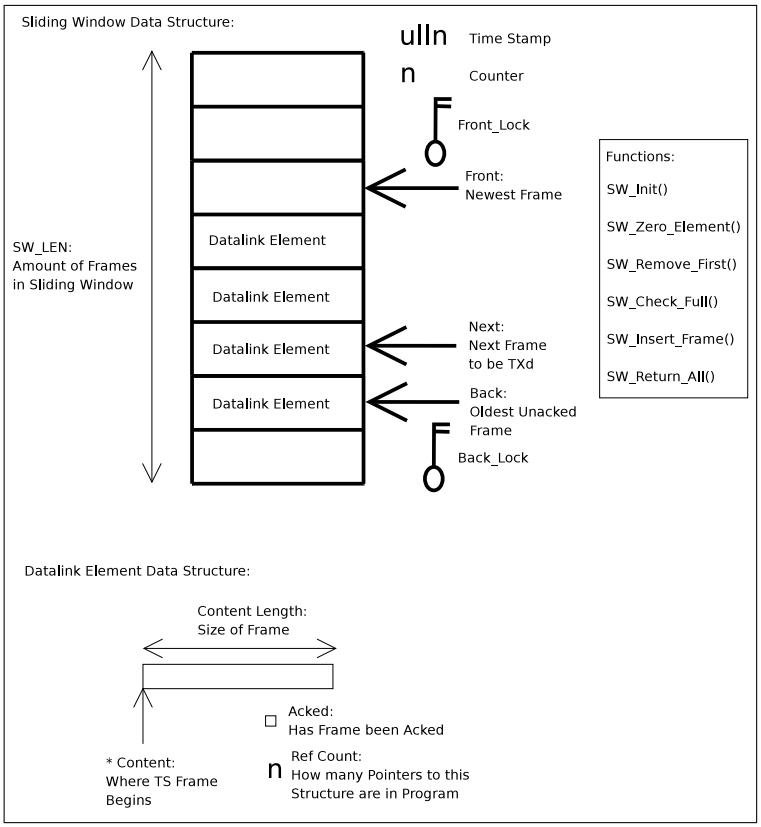


Figure 4.2.3: Sliding Window Data Structure Overview

The variables front and back are both accessed by both the check channel thread and the routine thread. Hence these two variables need to be protected by a mutually excluding lock (mutex). The front lock and back lock mutex are locked while the variables front and back are accessed.

The time stamp variable is used in place of a timer. The reason for this choice is the fact that the resetting of the next pointer is a non critical event that only needs to occur when the channel is serviced. The occasional comparison of a time stamp with the current time is computationally much cheaper than a dedicated timer.

The counter variable is used to keep track of the number of times the first frame has been transmitted since it became the first frame. This variable can be used to determine a lost link.

The Sliding Window Data Structure contains Data Element Data Structures. A Data Element Data Structure is a frame which has the following variables:

- * Content - Points to the location where the actual data is stored in memory

- Content Length - Indicates the length of the frame
- Acked - Indicates if an acknowledgement for this frame has been received
- Ref Count - Indicates how copies of the content pointer are in the program (if absent this is an expedited frame)

The ref count variable prevents a frame from being deleted while there still are pointers to the content in the program. The acked variable is used to indicate that the frame has successfully been received. When acked is one and ref count is zero the SW_Remove_First() function may be called to remove the frame.

4.2.2.2 Buffer Data Structure

The Buffer Data Structure is used to pass data between the TS and the SCCS, hence the data structure may be accessed simultaneously by different threads. The design has therefore to be thread safe.

The similarity between the Buffer Data Structure and the Sliding Window Data Structure would make inheritance a lucrative design choice. Unfortunately the C programing language does not support inheritance. The size of the Buffer Data Structure is a soft design choice which is lenient to generosity. This leniency is due to the fact that a Queue Element Data Structure requires space only for four integers. The Buffer Data Structure maintains the following variables:

- Back - The oldest occupied position
- Front - The next position to place data
- Back Lock - The back mutex
- Front Lock - The front mutex

The reason for the simplicity of variables used in the Buffer Data Structure is the fact that the data structure acts only as a conveyor belt between the individual sublayers. As such, only the front and back positions are required.

Queue Element Data Structures is enqueued on the Buffer Data Structure. The Queue Element Data Structure maintains the following variables:

- Content - Where the data begins
- Content_Length - The length of the data
- Channel - Which channel this data is associated with
- * Ref Count - A pointer to the Ref count in the case of data being a type AD frame or zero in all other cases

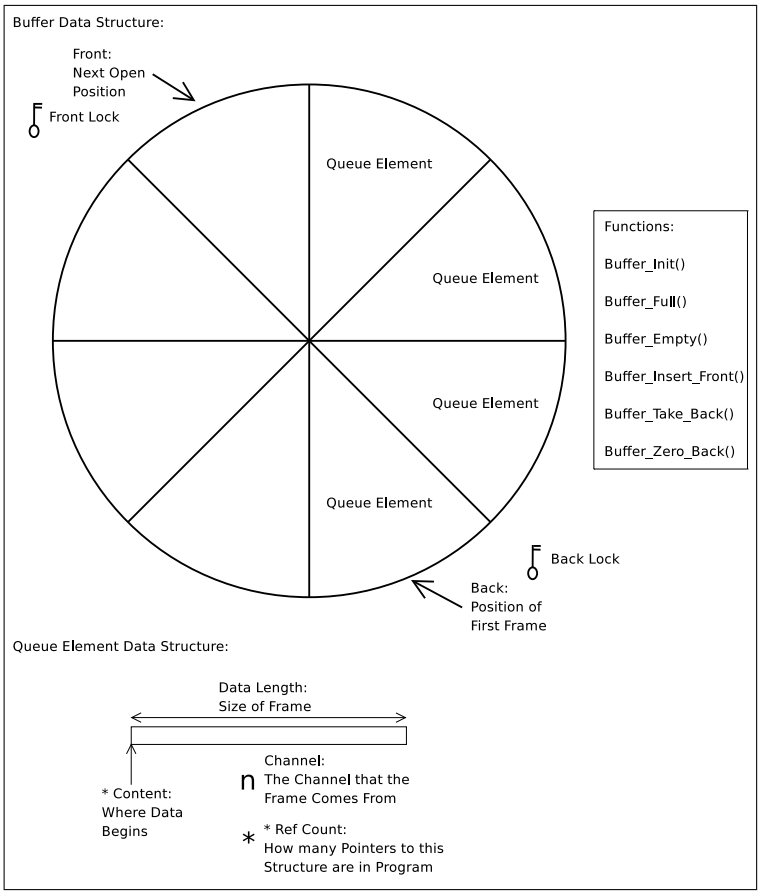


Figure 4.2.4: Buffer Data Structure Overview

Since the routine thread performs the multiplexing as well as demultiplexing of the data, the channel data needs to be explicitly stored. The ref count is used to determine whether the data to be transmitted should be regarded as reliable or not. In the case of unreliable transmission a frame is not associated with a frame sequence number, nor is it stored by the transmitting node.

4.3 Synchronisation and Channel Coding Sublayer

The SCCS fulfils the requirements of:

- accessing the physical medium
- adding coding gain by means of FEC
- keeping the synchronisation lock

- deliviering TS frames with best effort

The SCCS has two sets of functionality, which should run concurrently, these are reading to and writing from the physical medium.

4.3.1 SCCS Overview

The SCCS interfaces with the physical layer in the protocol stack. The physical layer is a radio, which is accessed through the expansion port of the FPGA. The expansion port is accessed via a file descriptor at location "/dev/exp".

The outgoing data segments are BCH encoded and synchronisation markers are added. Inbound data segments are stripped of their synchronisation markers and then BCH decoded. This is an attempt to correct any errors which may have been introduced into the data during transmission.

4.3.2 BCH

In Section 2.4.5 BCH codes, a type of FEC, were introduced. The great advantage in using BCH codes is their simplicity. Errors can be corrected with little computational effort. Hence the BCH encoding and decoding can both be done in software on the SH-4 microprocessor.

When deciding on the block length of a BCH code block there is a tradeoff. BCH codes are block codes, which means that only blocks of a particular designed length can be encoded and decoded. Hence, if a data block is shorter than the designed length, the block needs to be filled to the predefined length using filler data. An advantage of shorter code blocks is that less filler data is required to attain the designed length. Furthermore, in a short code block fewer errors tend to occur, which means the decoding is simpler. The advantage in using longer code blocks lies in the fact that the LCM of primitive polynomials is used to encode the code block. The amount of redundant data required to correct t errors in a code block of length $2^m - 1$ is less than twice that required to correct $\frac{t}{2}$ in a code block of length $2^{m-1} - 1$. The advantages of shorter code blocks, however, outweigh those of using long code blocks. The following sections, 4.3.2.1 and 4.3.2.2, discuss the encoding and decoding methods used in the protocol implementation.

4.3.2.1 Encoding

The minimal polynomial used to generate the code block was $\phi(x) = 1 + x + x^6$. This polynomial introduces a Hamming distance of at least 3 between code words. Thus all code words which have zero and one error will be decoded into the correct data block.

A simple method by which the Hamming distance can be increased is by adding a parity check. A parity check can be added by multiplying the data by a polynomial of $par(x) = x + 1$. This increases the Hamming distance

by one. By adding parity to the g(x) an extra error detection step is added between decodable code blocks. If a received code block is found to have two errors it is discarded as an uncorrectable code word. The Hamming distance and the resulting bit errors required to transform one data word to another is shown in Figure 4.3.1.

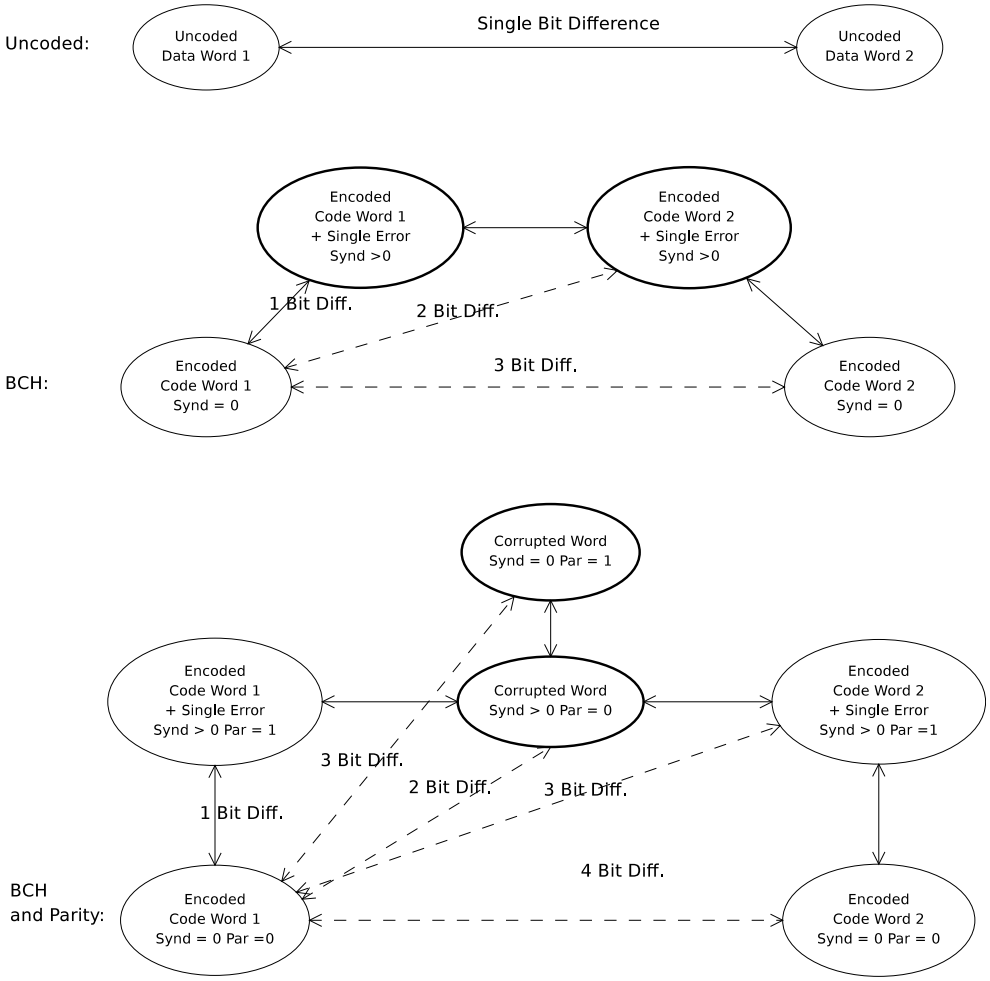


Figure 4.3.1: Hamming Distance Introduced by BCH encoding and Parity Check

Listing 4.1: C code for BCH and Parity Encoding

```
polynomial = 0x8A;
FEC = 0x00;
for (i = 7; i > 0; --i) {
        byte = data[i];
        for (j = 0; j <  8; ++j) {
                carry = (byte ^ FEC )& 0x80;
```

```
                    FEC <<= 1;
                    byte <<= 1;
                    if (carry){
                            FEC = FEC ^ polynomial;
                    }
            }
}
data[0] = FEC & 0xFE;
```

Listing 4.1 shows the code used to encode the data. The array "data[]" is 8 bytes long, of which 7 bytes contain data. The variable "FEC" is used as a dummy variable for the redundant bits. The generator polynomial, $g(x) = \phi(x) \cdot par(x) = 1 + x^2 + x^6 + x^7$, maps to the hexadecimal 0x8A. The following steps are repeated over the data:

1. modulo two add the front bit of the data to the front bit of the redundant bits.
2. shift data as well as redundant bits.
3. if the modulo two addition in step 1 results in a carry, add the polynomial to the redundant bits.

These steps effect the long-division of the data by the generator polynomial. Once the redundant bits have been computed they are added to the data and encoding is complete.

4.3.2.2 Decoding

Decoding involves determining the syndrome and the parity of the received code block. There are four distinct combinations of parity and syndrome. The combinations and the associated reasons and interpretations are shown in Table 4.3.1.

Table 4.3.1: Syndrome and Parity Check Evaluation

		Synd	Par	Interpretation	Reason
A	1	0	0	Correct	No errors
	2				An even number of at least 4 errors
B	1	>0	0	Correctable	A single error
	2				An odd number of at least 3 errors
C		>0	0	Uncorrectable	An even number of at least 2 errors
D		0	1	Uncorrectable	An odd number of at least 3 errors

The decoder interprets and corrects any single error. All cases with double

errors are detected and the code block is rejected. Of the code blocks containing 3 errors, some are rejected and some are falsely accepted. Of the code code blocks containing 4 errors, some are rejected and some are falsely accepted. The chance of a bit in a received code word being corrupted is $63 \cdot BER$. The chance of two bits being corrupted in the same code block decreases to $63 \cdot 62 \cdot BER^2$. Since the BER is usually a lot smaller than $\frac{1}{64-n}$ the chance of additional errors being present in a code block increasingly decreases.

The first step in decoding is determining the syndrome. Using the fact that the syndrome of a code block is equal to the modulo sum of the syndromes of each bit in the code word, the syndrome can be computed without much computational complexity. During the design phase an extensive list of syndromes was computed. This list was computed by a Python script, which is given in the Appendix B. The evaluated syndromes for each bit have been placed in a lookup table. The syndrome of the received code block can thus be determined by a series of value lookups and modulo two additions. The code used for this operation is shown in Listing 4.2.

Listing 4.2: C code for Determining the Syndrome

```
S1 = 0x00;
for (N = 0; N < 64; ++N){
        shift = 0x01 << (N%8);
        if (data[N/8] & shift){
                S1 ^= LOOKUP_H[N%63];
        }
}
```

Having determined the syndrome, the next step is calculating the parity of the code word. The combination of parity and syndrome is then used to interpret the state of the case of the received code block. If the code block is thought to contain only one error and can therefore be corrected, the error location is determined.

Determining the error location is the final step in correcting a code block deemed correctable. Once the error location is found, the corresponding bit can be flipped and thereby the code block is corrected. The method in which the error location is found is again by means of a lookup table.

Since the decoder can correct up to one error, there are 63 error location polynomials. Each of the 63 error location polynomials has only one term. A single erroneous bit at any specific location in the code block always results in the same syndrome, regardless of the underlying arrangement of the original data. The one to one mapping of the error location numbers and the syndrome values can be reversed to determine the error location for any given syndrome. The syndromes resulting from the error locations were determined using a Python script. The resulting relationships between error location numbers and syndrome values are given in Appendix B. The error location numbers were then arranged in ascending order of the correspon-

ding syndromes. The new arrangement of error location polynomials was placed in an array, which was used to look up the error locations.

In the software the error location array, "ERROR_L[]", will have the error location number corresponding to the syndrome, "S1", at location "S1-1". The value obtained from the array "ERROR_L[]" is then used to determine which bit should be flipped. The corresponding bit is then flipped.

This section has discussed the implementation of a single error correction BCH code with added error detection capabilities.

4.3.3 Communications Link Transmission Unit

The Communications Link Transmission Unit (CLTU) introduced in Section 2.6.3.3 and shown in Figure 2.6.11 is the data unit, which is transferred between the sending and receiving ends of the SCCS. The 64 bit encoded data cells, as described in Section 4.3.2.1, contain the payload data. The data is preceded by a start sequence and terminated by a stop sequence.

The receiving end of the SCCS is in either the searching or the decoding state. While the receiving end is in the searching state it parses incoming data looking for a start sequence. The searching state is only left if an acceptable start sequence is detected. This includes a start sequence with a single error.

While the receiving end is in the decoding state, each following 64 bit pattern is treated as encoded data and decoded. If the stop sequence is detected, then the receiving end enters the searching state.

The start sequence is used to maintain synchronisation between the sending and receiving nodes. The stop sequence signals the end of a frame. The stop sequence is an uncorrectable code block. The stop sequence is of type D in Table 4.3.1.

4.3.4 RX Thread

The functionality of reading from the physical medium is performed by the RX thread. The physical layer is accessed at the file location "/dev/exp". The file location "/dev/exp" maps to the expansion port. The flow diagram of the RX thread is given in Figure 4.3.2.

After initializing the required variables, the thread blocks until new data is available on the physical medium. At this point in the program data passes from the lower layer to the datalink layer.

If the incoming data is for a channel which is in the searching state, then the new data is scanned for the start sequence. Once the start sequence is located, the SCCS enters the decoding state.

If new data arrives at the receiving node and the SCCS is in decoding state, then the new data is decoded. If this data is the first code block then a buffer of appropriate length is created and the data is added to this buffer.

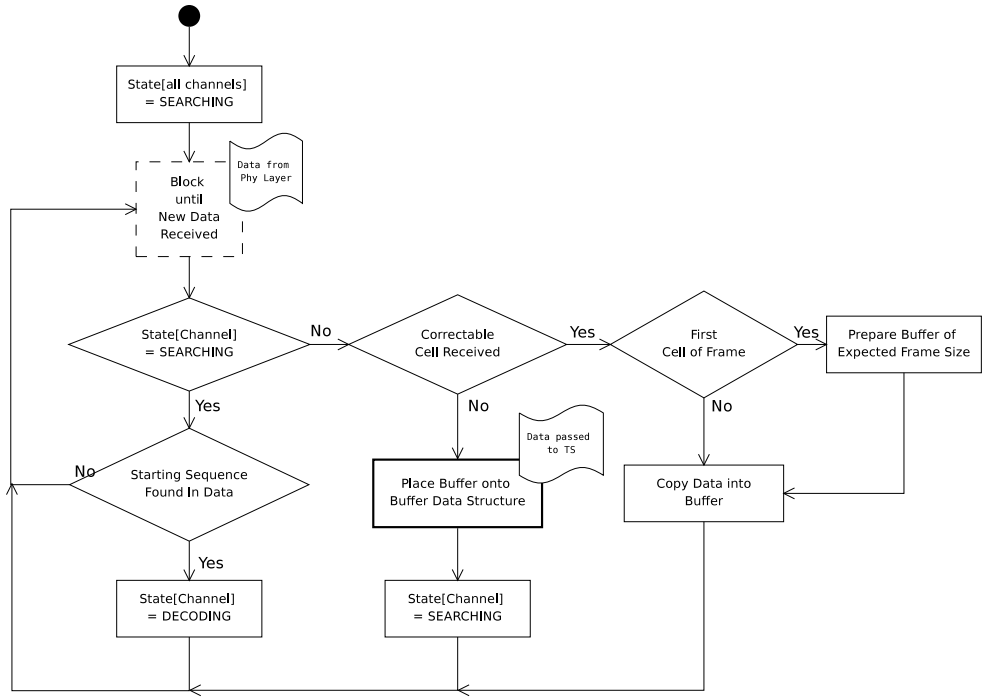


Figure 4.3.2: SCCS Receive Thread Flow Diagram

All subsequent, correct and correctable incoming code blocks are appended to this buffer. When a uncorrectable code block arrives the buffer is passed to the RX buffer data structure and the searching state is entered. At this point data is passed from the SCCS to the TS. The RX thread then reenters the searching state.

4.3.5 TX Thread

The functionality of writing to the physical medium is performed by the TX thread. The flow diagram of the TX thread is given in Figure 4.3.3.

After initializing, the TX thread blocks while there is no data to be transmitted. If new data is available the new data is taken from the TX buffer data structure. At this point data is taken from the TS.

The transmission is initiated by the transmission of the start sequence. Next, the received data block is broken into 7 byte chunks, each of which is encoded into 8 byte cells. If the final data chunk is less than 7 bytes, filler data is appended to it. When the last data chunk has been transmitted, the stop sequence is transmitted. In each of the preceding steps data is sent to the physical medium.

If the data received from the TX buffer structure has a "Ref count" pointer, then this implies that the a type AD frame, which needs to be acked, has been

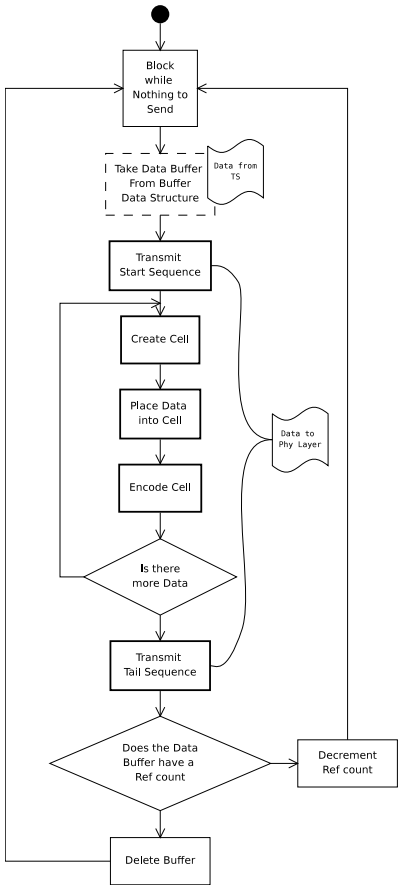


Figure 4.3.3: SCCS Transmit Thread Flow Diagram

transmitted. If there is no frame then an expedited frame or a CLCW has been transmitted.

A type AD frame should only be deleted if a CLCW acks the frame and there are no pointers to the frame in the datalink layer. The "Ref count" is decremented to indicate that one less pointer to the frame is in the datalink program.

A CLCW and an expedited frame are both unreliable. Therefore when a CLCW or an expedited frame has been transmitted, the data should be deleted.

4.4 Transfer Sublayer

The TS fulfills the requirements of:

- delivering SS segments correctly

- delivering SS segments in sequence
- multiplexing the SS channels into a single seqence of frames

None of the functionality required in the SS requires concurrent execution, thus the functionality was implemented in a single thread.

4.4.1 TS Overview

The TS is the heart of the TC-SDLP. The following description gives an overview of the operations which are performed in the TS.

The TS provides error free and in sequential transfer of data between the sender and receiver. The mechanism ensuring the reliability is called COP-1. At the sending end the incarnation of COP-1 is known as FOP-1 and at the receiving end it is known as FARM-1. The COP-1 mechanism makes use of a sliding window to ensure correct sequencing of frames.

The states of the various aspects of COP-1 are communicated by means of headers at the sending end and in CLCW at the receiving end. This header procedure happens below the COP-1.

Frame correctness is verified by means of a CRC check. Appending the CRC check is the last process which the sending end performs before passing a frame to the next lower sublayer. At the receiving end checking the frame correctness, by evaluating the CRC, happens after any filler bits have been removed.

4.4.2 COP-1

COP-1 is the mechanism that ensures data transfer that is in sequence and error free. The sending end incarnation of COP-1 is FOP-1 and the receiving end incarnation FARM-1.

4.4.2.1 FOP-1

FOP-1 is the mechanism that controls the sequencing of the TS at the sending end. Data to be transmitted is placed within a TS frame, of which the structure is discussed in Section 4.4.3.

The FOP-1 mechanism manages the sliding window, which is detailed in Section 4.4.2.3. FOP-1 is a state machine which is described by means of a state table as well as state transition diagram. The detailed state table spans seven pages and was therefore excluded from the body of the thesis. The state diagram may be found in Figure 4.4.1. The states are:

S1 Active

S2 Retransmit without wait

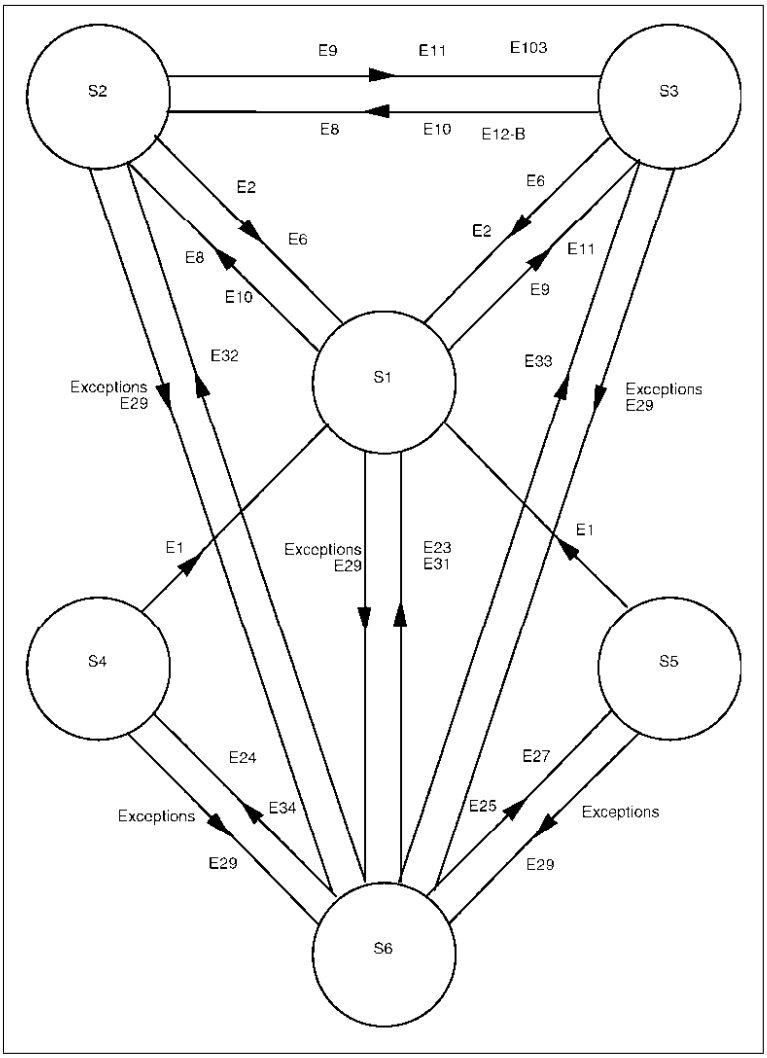


Figure 4.4.1: FOP-1 State Diagram

S3 Retransmit with wait

S4 Initialising without BC frame

S5 Initialising with BC frame

S6 Initial

4.4.2.2 FARM-1

FARM-1 is the mechanism that monitors the arrived frames at the receiving end. Every received frame is evaluated according to what was received versus what was expected. The FARM-1 mechanism then creates an ACK or

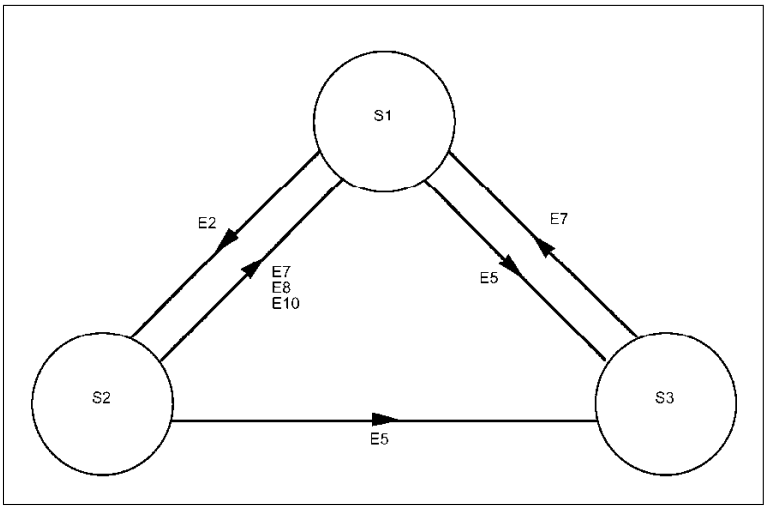


Figure 4.4.2: FARM-1 State Diagram

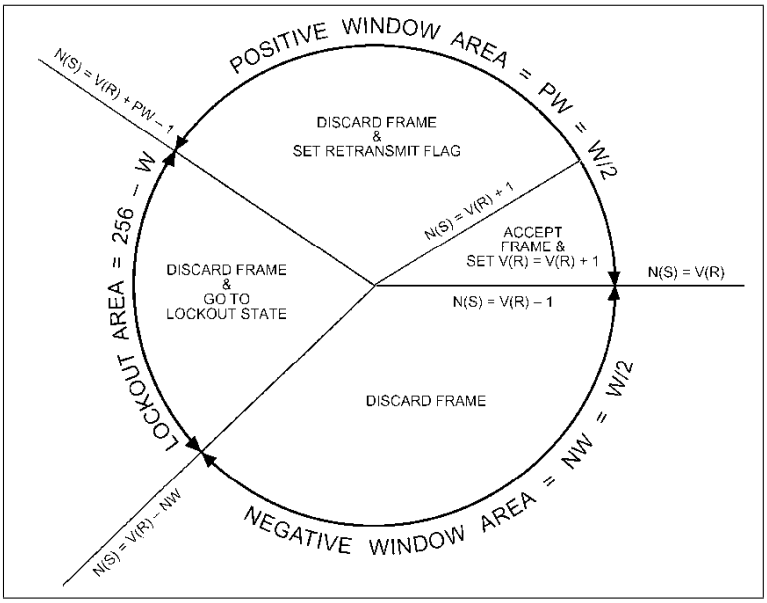


Figure 4.4.3: FARM-1 Sliding Window Concept

negative acknowledgement (NACK). These ACKs are contained in reporting frames known as CLCWs. The structure of the CLCW is discussed in Section 4.4.3.

The go-back-N sliding window mechanism, as described in Section 4.4.2.3, is used in the FARM-1 mechanism. The concept of the sliding window, as depicted in Figure 4.4.3, is used to make a decision about the sequence of the

current type AD frame. The region within which the sequence number of the received frame falls, with regard to that which is expected, triggers various actions.

The detailed actions of the FARM-1 mechanism, as was the case with FOP-1, are contained in a state table spanning several pages. The state table has thus been excluded from the body of the thesis but the state transition diagram is shown in Figure 4.4.2. The states are:

S1 Open

S2 Wait

S3 Lockout

4.4.2.3 Sliding Window

The sliding window mechanism was introduced to take advantage of duplex links that would remain idle in the interval of transmitting a frame and receiving an acknowledgement for that frame. A unique number is assigned to each frame in the sliding window mechanism. In this way an acknowledge for that specific frame can be identified.

The TC-SDLP makes use of the go-back-N sliding window mechanism, which ensures reliability by maintaining a sequence of unacknowledged frames up to the "sliding window length". The frames are removed only if they have reached the front of the queue and have been acknowledged. The sequence number uniquely identifying each of the frames in the sliding window queue can thus stem from a number pool of at least twice the size of the sliding window length.

To determine the customization of the sliding window mechanism the two situations of normal and erroneous operation should be investigated. Under normal operation there should be enough frames in the sliding window to ensure that there is no idle time waiting for ACKs. The correct sliding window length value can thus be determined from an investigation of the normal operation. A sequence of lost frames and acks could result in an error situation, in which the transmitting end does not receive a single acknowledgement nor acknowledgement. Such a situation results in the transmitter remaining in a waiting state which can only be exited by means of an internal trigger.

The length of a sliding window is determined by investigating the normal operation of the sliding window. Under normal operation the frame size may vary between 16 and 1024 bytes. After adding FEC and the start and stop markers, the encoded frames range between 208 and 9488 bits. At a baud rate of 9600 bits/sec the number of bits correspond to $21.\dot{6}$ and $988.\dot{3}$ ms. Between a bit being sent and it arriving there is a propagation delay caused by the time taken for the electromagnetic signal to travel in space. At the

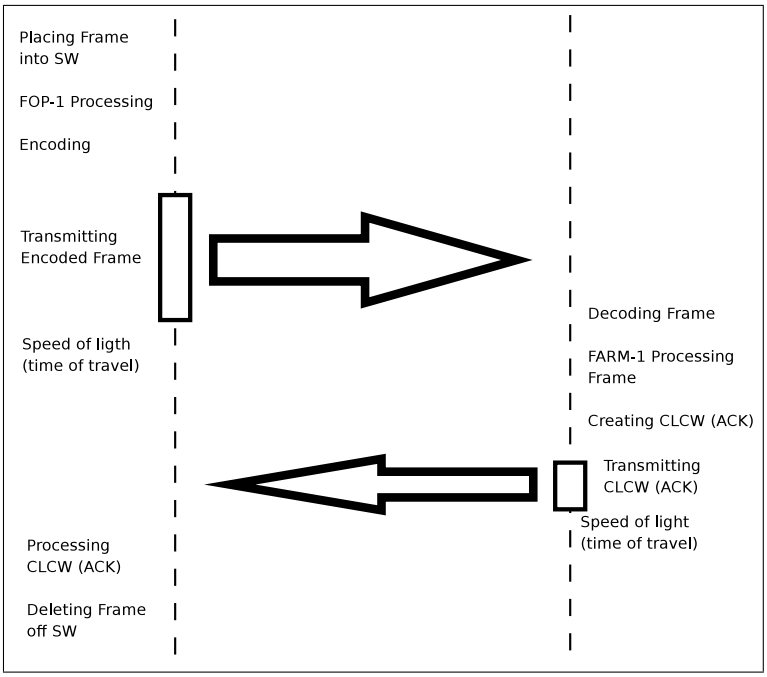


Figure 4.4.4: TS Routine Thread Flow Diagram

maximum distance between a ground station and the satellite, as described in Section 3.2.2, the propagation time is 17.17ms. An acknowledge has a fixed size and, once encoded, occupies 144 bits and takes 15 ms to transmit. Finally the processing time, comprising operations such as encoding and decoding has to be accounted for. Since accurately measuring all these processing times is, at best, difficult, a contingency is added into the time equation. The total time budget for a maximum length size is the time taken as given in Equation 4.4.1.

$$\begin{aligned} t_{max} &= t_{maxframetx} + 2t_{prop} + t_{ack} + t_{10\%contingency} \\ t_{max} &= t_{maxframetx} + 153.04 \end{aligned} \tag{4.4.1}$$

A sliding window mechanism should ensure that data is continually being transmitted. So during the 153.04 ms, which is the time span between transmitting the last frame bit and receiving the last ack bit, the transmitter has to be busy. A case in which a maximum length frame is followed by only minimum length frames, will result in $\frac{153.04}{21.67} = 7.06$ frames being transmitted. The sliding window length should therefore be at least 9 frames long.

The chance of a state being entered, from which a timer reset is required to recover the operations is rare. The timer should not impact the normal operation of the protocol and was therefore chosen to encompass the successful transmission of three maximum length frames, 3,424 s. The timer was sim-

plified to a time stamp which adds 3.424 s to the current time. These values were revised to accommodate the slower baud rate in the internal loop-back (Section 5.2.1).

4.4.3 Frame and CLCW Structure

The frame sturcture of the TS frame is given in Figure 2.6.9. The TS frame header contains the variable values which FARM-1 requires. The most important of these variables in the frames are discussed below.

At bit positions 2 and 3 the frame type is recorded. The frame type indicates whether the frame carries payload data or configuration messages and whether it is expedited. Expedited frames are not transfered any faster than normal frames, but they are unacknowledged.

At bit positions 24 through 31 the length of the frame is given in bytes. The space lost in these 10 bits is the tradeoff for having a variable length frame. The frame can hence have a maximum length of 1024 bytes.

At bit positions 32 through 39 the frame sequence number is given. These 8 bits offer up to 255 unique sequence numbers.

The CLCW ACKs frames both positively and negatively.

4.4.4 CRC

The CRC, as described in Section 2.5.5.2, is an ECC, which uses Galios Field mathematics to ensure the successful detection of almost all errors. a CRC-16 check is employed in the protocol. The redundant bits of the CRC are appended to the TS frame. The location of these bits is at position $5 + DataLength$ and spans two bytes.

The code used to create the CRC redundant bits free may be found in Listing 4.3. The redundant bits obtained from the code in Listing 4.3 are appended to the TS frame. If the newly encoded frame is subjected to the code in Listing 4.3, the resulting redundant bits will evaluate to 0. Hence the frame is checked for errors by the same code as is used to generate the redundant bits. If, after error checking has been completed, the "crc" integer of Listing 4.3 evaluates to the a non zero value, then frame is known to have been corrupted.

Listing 4.3: C code for determining Redundant bits for CRC

```
crc = 0xffff;
for (i = 0; i < length; ++i) {
        byte = data[i];
        for (j = 0; j < BYTE_LEN; ++j) {
                carry = ((byte ^ crc) & 0x01);
                crc >>= 1;
                byte >>= 1;
                if (carry) {
                        crc = crc ^ GX;
```

```
                    }
                }
}
```

4.4.5 Routine Thread

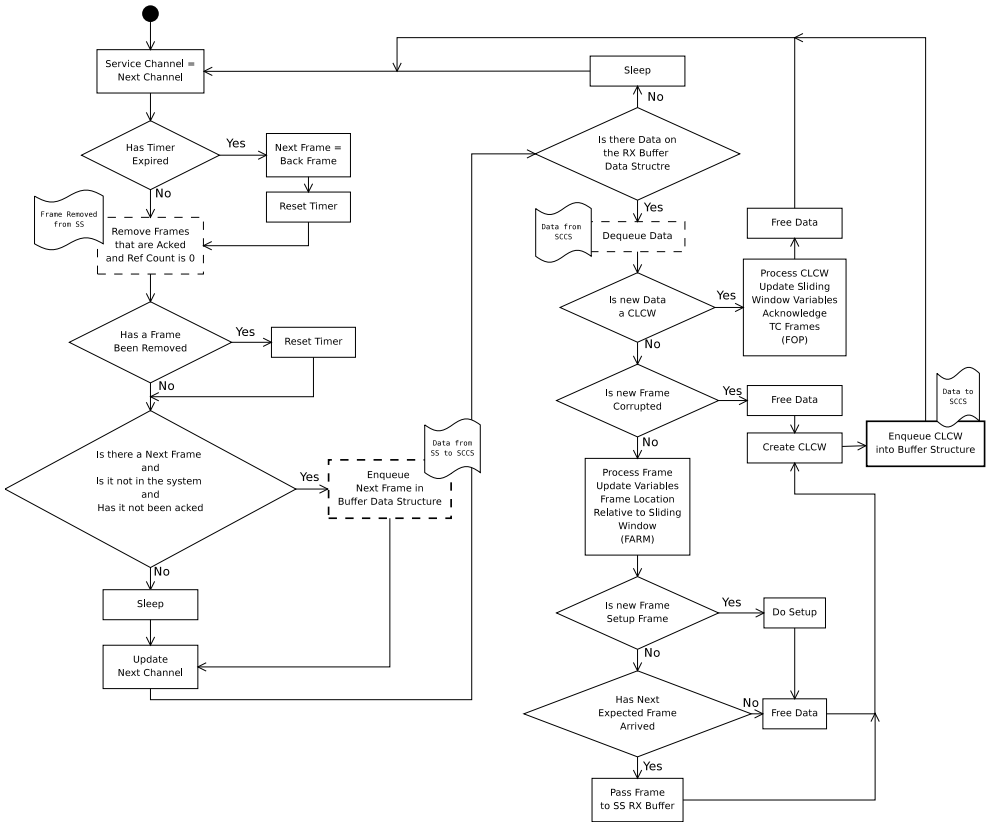


Figure 4.4.5: TS Routine Thread Flow Diagram

The routine thread is by far the largest part of the program. The routine thread encompasses both the FOP-1 and the FARM-1 state machines of COP-1. Furthermore, since only one expansion port is used to send and receive data, through multiplexing is required. The routing thread multiplexes and demultiplexes the data. Lastly, the routine thread is required to interface with the SCCS and the SS.

The interface between which the SS passes data to the FOP-1 component of the routine thread is the Sliding Window Data Structure. The manner in which the various locks are applied in order to ensure no simultaneous attempts at accessing a single data piece may be found in the discussion of the

Sliding Window Data Structure, Section 4.2.2.1. Data is passed between the routine thread and the SS by means of a single space for a single frame. This memory space is also duly locked to ensure single access to the data. The data are passed between the TS and the SCCS by means of the TX Buffer incarnation of the Buffer Data Structure. The description of the generic Buffer Data Structure may be found in Section 4.2.2.2.

The routine thread is a sequential program. The flow diagram outlining the routine thread is given in Figure 4.4.5. Once all internal variables of the routine thread have been initialized, the next channel to service is selected. The next channel is then serviced. Servicing a channel involves checking and resetting the timer of the channel, removing all acknowledged frames and, if the next to be transmitted frame in the Sliding Window Data Structure meets all requirements, enqueuing this frame in the Buffer Data Structure. At this point data is thus passed from the sending end of the SS to the TS and then to the SCCS. It must be noted that the data is not physically copied to the Buffer Data Structure, but that a pointer to its position in the Sliding Window Data Structure is placed on the TX Buffer. By using a single Buffer Data Structure between the transmitting end of the TS and SCCS, multiplexing is implicitly achieved.

Once the channel has been serviced, the next channel is updated. The channel is updated by a circular incrementing macro:

```
CIRCULAR_INC(var,CONST) ((var) = ((var) +1  == (CONST)) ? 0
    : (var) +1)
```

This shorthand increments over a Finite Field of length "CONST", where const in this case is set to the number of channels. This and other similar macros also aid in the arithmetic and checking of other finite size data structures.

Next, the routine thread checks if any new data has arrived on the RX Buffer. If nothing has arrived, the routine thread sleeps and then attempts to service the next channel. If data has arrived it is dequeued, hence receiving data from the SCCS. The arrived data type is determined and if a CLCW has arrived, it is processed and all relevant FOP-1 state transitions and variable updates are performed. Then the routine thread restarts and the next channel is serviced.

If a frame has arrived, the integrity of the frame is determined. In the case of an error bearing frame being received, the frame is discarded. If an error free type AD frame is received the relevant FARM-1 state transitions and variable updates are performed and the frame is passed to the SS. If the frame is a setup frame, then the relevant setup is performed. Finally, a CLCW is generated. The routine thread then restarts and the next channel is serviced.

4.5 Segmentation Sublayer

The SS fulfills the requirements of:

- Creating an interface through which the next higher layer can access the datalink program
- Delivering a file stream with best effort correctness
- Segmenting the file stream into frames which adhere to limits imposed by the TS
- applying the optimal segmentation length strategy presented in Chapter 3.

4.5.1 Segmentation Sublayer Overview

The SS creates the interface by which the client programs access the service which the datalink layer offers. The client programs may open a channel to read or to write.

The SS at the transmitting end breaks the file to be transmitted into data segments which are no greater than the maximum size that the TS accepts. At the receiving end the segments are reassembled. Hence a virtual file streaming connection is set up. The size of the segments is optimally chosen according to the estimation method presented in Section 3.5.

The SS offers the interface by which the next higher layer can access the datalink layer. The interface can be opened in either read or write mode, depending on whether the device is at the transmitting or receiving end of the communication.

In read mode the data interface remains open for the duration of one file received, or until a timeout occurs. The Segmentation Sublayer headers contain information about the current segments position in the file. The "last segment (10)" or "only segment (11)" header identifiers are used to notify the end of file and the Segmentation Sublayer closes the session after such an identifier is received. Up until such a point the Segmentation Sublayer writes the received segments to the client program.

In write mode the client program writes a file to the Segmentation Sublayer. The file length is compared to the optimal segmentation length. If the file, or the remaining part of the file, is smaller than the optimal segmentation length, the file, or the remaining part of the file, is encapsulated in one segment. If the file, or the remaining part of the file, is greater than the optimal segmentation length, a part of the file, or the remaining part of the file, equal to the optimal segmentation length is encapsulated in a segment.

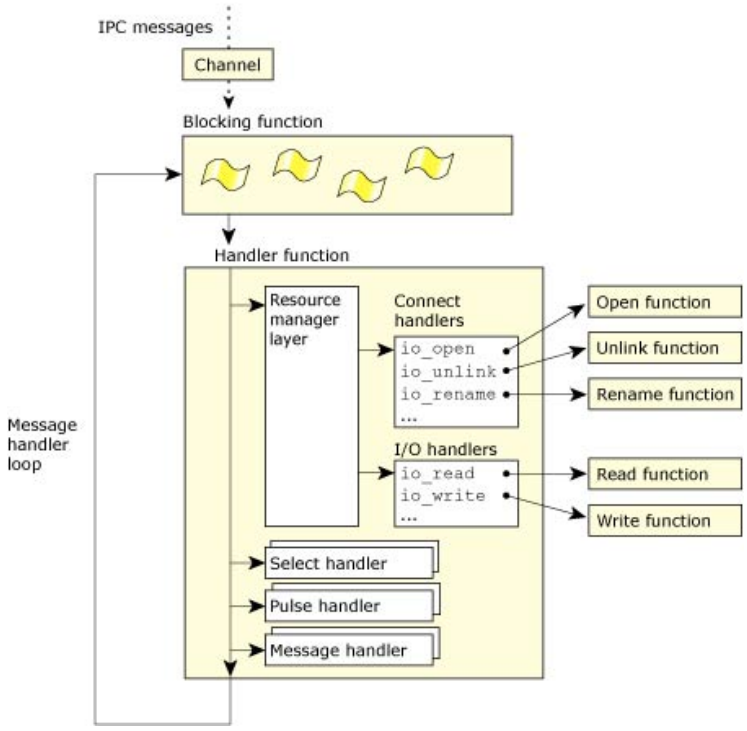


Figure 4.5.1: Dispatch Layer of Recourse Manager

4.5.2 The Datalink Layer as a QNX Resource Manager

QNX is a fully POSIX compliant OS. It uses POSIX calls to read and write the calls interface within the OS. It is therefore best practice to extend the POSIX interfacing method to the datalink program. The QNX "resmgr.h" library offers a resource manager framework, which attaches functions to calls on a file descriptor associated with a file.

Hence an effective method of offering a resource, in this case the datalink layer, to the rest of the system via a file pointer. In QNX terms the datalink protocol program is a resource manager. The file locations "/dev/ch0" through "/dev/ch*N*". Calls to these locations are handled by the dispatch layer. The conceptual ordering of the dispatch layer is shown in Figure 4.5.1. Common functionality on each of these functions is defaulted and where necessary it is extended or replaced.

The open function was extended to include a control setup message to be sent to the receiving node, resetting internal variables such as those associated with sequencing. The write function was replaced by a function to convert the incoming file to segments which are passed to the TS. The read function was replaced by a function which passes incoming segments as a reassembled file to the client program.

The code used to create the association is presented in Listing 4.4. Each channel to be opened receives a new thread within which to operate. This allows for each channel to run smoothly without being affected by the operation of another channel. The last channel gets the current thread to operate in.

Listing 4.4: C code for Attaching File Descriptors with SS Functions

```
memset(&resmgr_attr, 0, sizeof(resmgr_attr));
resmgr_attr.nparts_max = 1;
resmgr_attr.msg_max_size = 1016;
file_type = _FTYPE_ANY;
flags = 0;
iofunc_func_init(_RESMGR_CONNECT_NFUNCS, &connect_funcs,
    _RESMGR_IO_NFUNCS, &(io_funcs));
io_funcs.read = channel_read; // This is the SS read
    function
io_funcs.write = channel_write; // This is the SS write
    function
connect_funcs.open = channel_open; // This is the SS open
    function
for (Channel = 0;Channel<CHANNELS;Channel+=1){
        if((dpp[Channel] = dispatch_create()) == NULL){
                printf("Unable to allocate dispatch handle.\
                    n");
                return EXIT_FAILURE;
        }
        devattr[Channel] = (struct device *) malloc(sizeof(
            struct device));
        devattr[Channel]->dev = dev;
        devattr[Channel]->Channel = Channel;
        iofunc_attr_init( &(devattr[Channel]->attr), S_IFNAM
            | 0666, 0, 0 );
        sprintf((char *)path,"/dev/ch%d",Channel);
        resmgr_attach(dpp[Channel], &resmgr_attr,(const char
            *) path, file_type, flags, &connect_funcs, &
            io_funcs, devattr[Channel]);
        if (Channel != (CHANNELS-1)){ // if not last channel
             give it a new thread
                pthread_create(NULL, NULL, CheckChannel, dpp
                    [Channel]); // Start RX thread while not
                    shutdown 1
        }
        else{ //last channel gets current thread
                CheckChannel(dpp[Channel]);
        }
}
```

4.5.3 Optimal Segmentation Length Calculation

The datalink interface may receive a file which is larger than the maximum size of a Segmentation Segment. In this case the file is to be truncated at a length convenient to encapsulate the new segment into a Segmentation Sublayer segment. The notion of the optimal segment length was developed and formulated in a previous Section 3.4. The optimal segment length can be calculated by Equation 3.4.2.

The optimal segmentation length calculator needs to have a knowledge of the history of failed frames. By means of a simulation it was discovered that a history length of 50 would result in a satisfactory estimation of the FER. The history was contained in the bits of 12 bytes. These 12 bytes contain the knowledge of the preceding 96 Frames. 96 Frames was, in Section 3.4.3, shown to be the optimal number of frames to use to determine what the best FÊR of the true FER is. Each time a segment is processed an estimation of the success of that frame is made. This is according to model 3 of Section 3.5. Although the Transfer Sublayer receives an ACK for each successfully transmitted frame and thus knows whether it has succeeded or not, this information cannot be accessed by the Segmentation Sublayer. If the Segmentation Sublayer were to access this information it would compromise the independence of the various sublayers.

The amount of time which should be afforded each frame depends on the segment that is being transmitted. Since it is too cumbersome to keep track of each frame's length, it is assumed that the current frame's length is similar to that of the frame at the bottom of the sliding window queue. Hence an estimate of how long the frame should take is based on the current frame's length. This may at times cause inaccuracies, but these are a worthwhile trade off not to have to keep track of information of the TS in the SS. For more detail on values and results, refer to Section 5.3.

4.5.4 Check Channel Thread

Each channel has a Check Channel thread associated with it. The starting of these threads was explained in Section 4.5.2. When a channel is opened the open routine is called, after which depending on the mode in which it was opened in, the read or write routine is executed. In the following sections 4.5.4.1, 4.5.4.2 and 4.5.4.3 these routines are elaborated on.

4.5.4.1 Open Routine

The open function is used to initialize the system. The sequence numbers are synchronised and the internal variables reset.

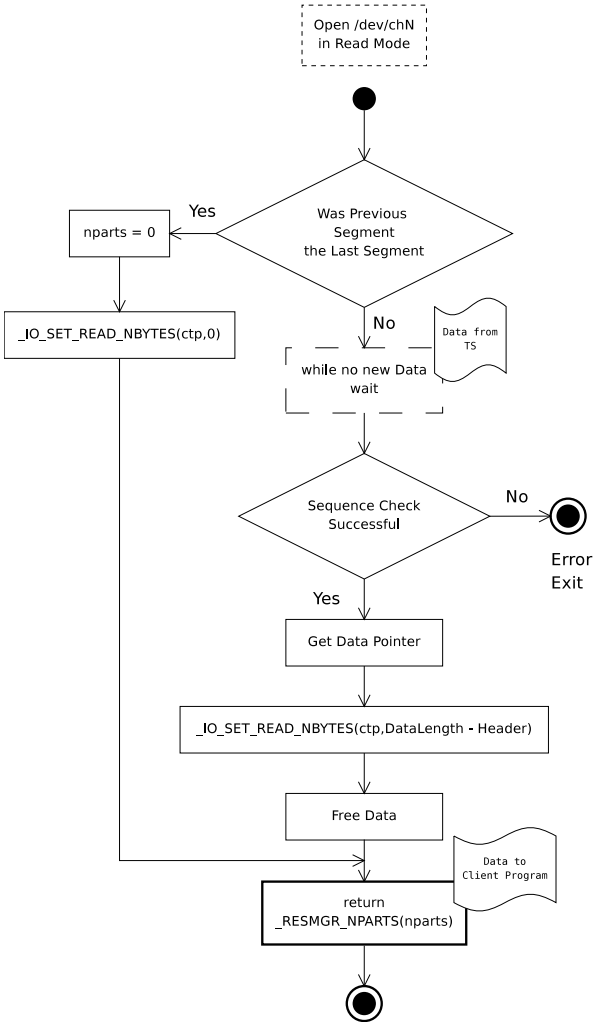


Figure 4.5.2: Segmentation Sublayer Read Routine

4.5.4.2 Read Routine

The read routine is used when the device is opened by the receiving node. The read routine is outlined by the flow diagram of Figure 4.5.2. The read routine is repeatedly executed until the channel is either externally closed or the final segments are transmitted. The routine starts by checking if the previous segment was the last one. If so, then a blank data block is returned to indicate this to the client program. If the previous segment was not the final segment, then the routine blocks until data is made available to it by the TS. Once the data has been received its sequence bits are investigated. If these sequence bits are illogical, a big error has occurred. The header is otherwise stripped off and the payload data is passed to the client program.

4.5.4.3 Write Routine

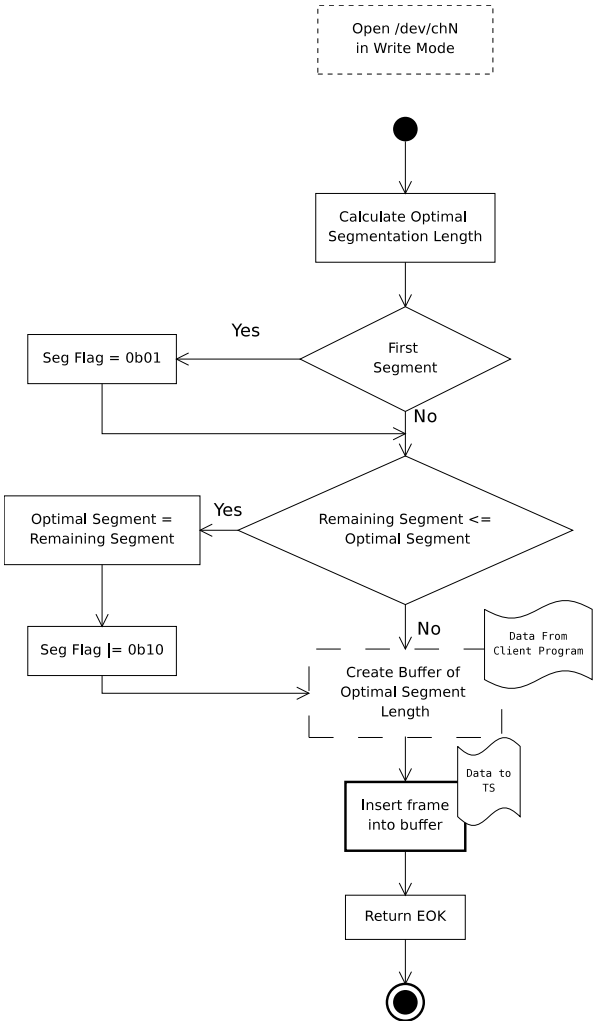


Figure 4.5.3: Segmentation Sublayer Write Routine

The write routine is used when the device is opened by the sending node. The write routine is outlined by the flow diagram of Figure 4.5.3. The write routine is repeatedly executed until either the device is externally closed or the file is completely transmitted. The write routine begins by calculating the current optimal segment length based on the FÊR. Next, if this is the first segment, the sequence bits are set accordingly. Next, if the remaining portion of the file is smaller or equal to the optimal segment length, then this is the last segment. If this is the case the corresponding bit of the sequencing bits is set. Then a chunk of the file either equal to or less than the optimal segment

length is copied into a buffer. This is where data is taken from the client program. Next the buffer is placed on the Sliding Window Data Structure. Thereby data is handed to the TS.

4.6 Logging

Logging is a process of storing certain soft data in order to describe a run in terms of certain units of measure. Logging is done at the data entry points and at the data exit points. Also in within FOP-1 the FER is measured. The logged data is presented in Chapter 5.

4.7 Conclusion

Chapter 4 has elaborated on the implementation of the TC-SDLP in software. The main data structures have been described. The sublayers and their implementation have been shown. The results obtained by logging the TC-SDLP implementation are given in the next chapter, Chapter 5.

Chapter 5

Results

This chapter presents results obtained from the simulation model as well as from the implementation of the TC-SDLP.

The final optimal segmentation model, Model 3, requires accurate estimation of frame failure based on time delay. The first set of results concentrates on the preciseness of time based error estimation. Next, the throughput obtained from the simulation is obtained as a benchmark. Finally the results from the system are presented and compared to the benchmark.

5.1 Benchmark Results

The simulation, presented in Section 3.6, was used to set a performance benchmark. At the various BER, the mean effective data throughput mean values and the corresponding standard deviation were obtained. These values may be found in the Appendix in Table C.1 as well as in Figure 5.1.1.

The standard deviation of the readings is very small, with maximum value of 0.000196, which shows that the results obtained from various simulation runs are sufficiently long.

Simulation results of effective throughput are very close to the expected effective throughput obtained from Equation 3.3.5 of Model 1. The accuracy of the simulation results shows, that Model 2, with a frame error history of 96, gives a precise estimate of the $\hat{FER}$. This in turn ensures that the optimal segment length is obtained.

5.2 System Test

The TC-SDLP implementation test setup included an internal loop-back test. The internal loop-back test was setup as a data loop within the SH-4 to collect the data presented in this chapter. For reasons named in Section 5.2.2, a full loop-back test was not possible.

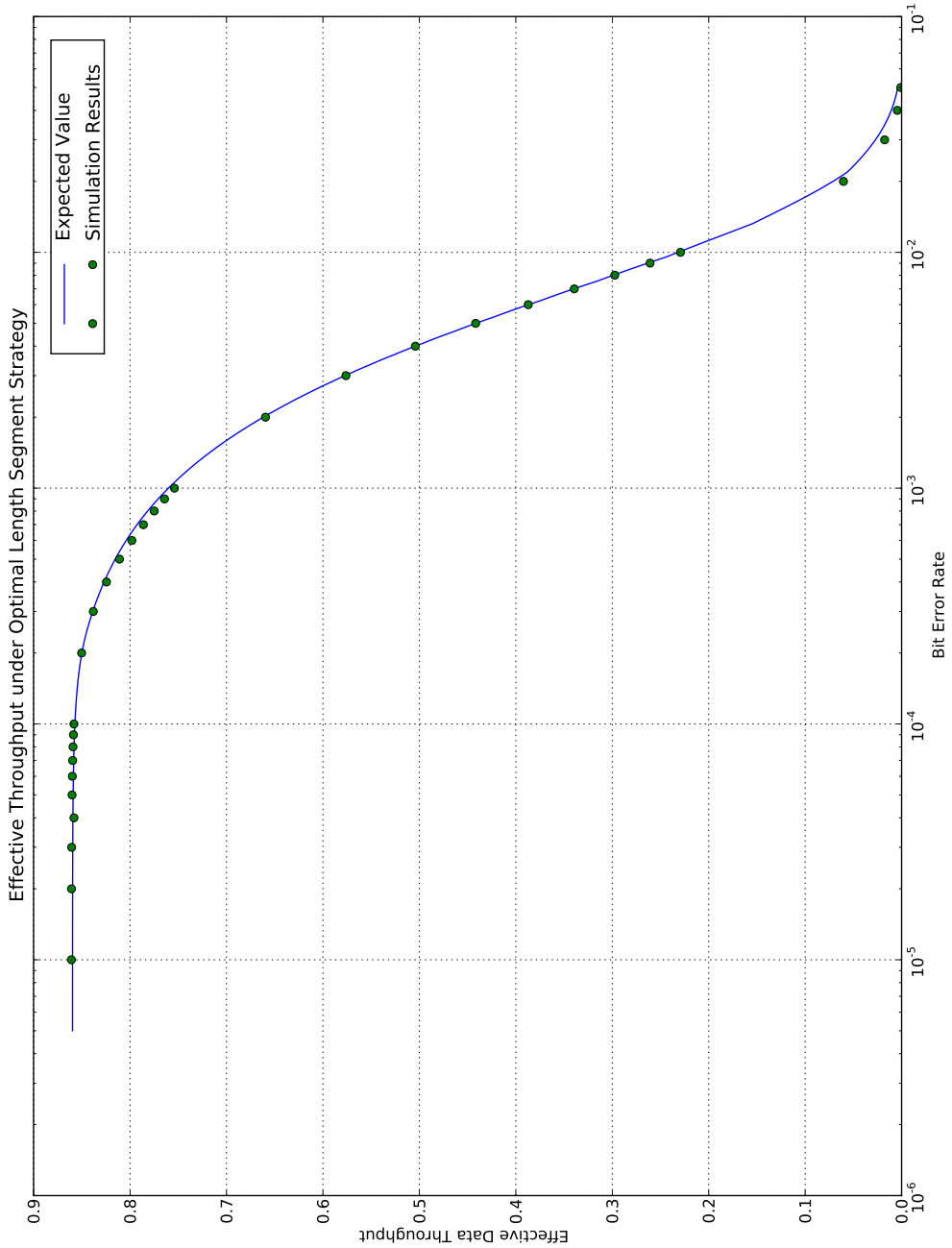


Figure 5.1.1: Simulation Results and Ideal Average Effective Throughput

5.2.1 Internal Loop-back

An internal loop-back was conducted within the SH-4 CPU. The transmitting end writes to a pipe and the receiving end reads from the same pipe. This setup demonstrates the functioning of the protocol, since all lower hardware and software layers are excluded. The pointer to the pipe, which was used as the data FIFO, is kept in the application wide variable "dev". Figure 5.2.1 shows the data flow of the internal loop-back. Errors were introduced by means of a the "Corrupter()" function and the channel numbers were inverted, i.e. channel 1 data was mapped to channel 3 and vice versa. The Corrupter() essentially compares a random number, generated by drand48(), to the BER. Due to the processing time causing delays, the baud rate was slowed down 2400.

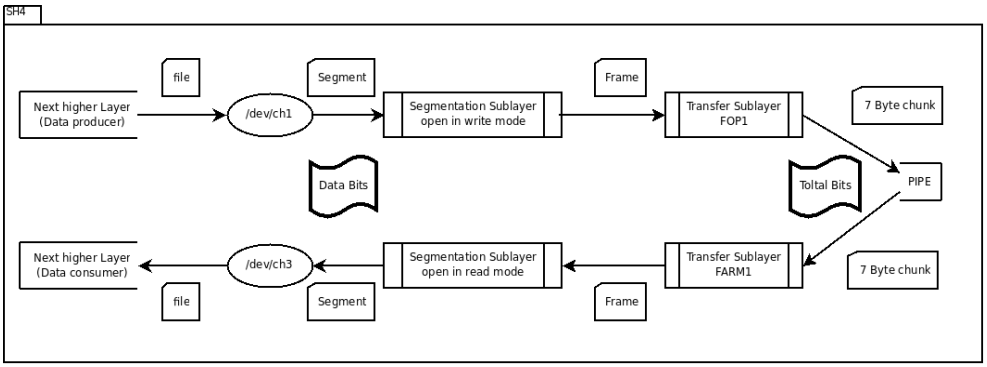


Figure 5.2.1: Internal Loop-back

5.2.2 Full Loop-Back

The full look-back test was not possible, because of an underlying problem in the Altera ProASIC FPGA's Dual Port FIFO architecture. A screen-shot of the Libero IDE informing users of this is given in Appendix D.0.1. The core has since been obsoleted. A workaround is in progress, but the development of this falls outside the scope of this work. A full loop-back test would involve a noise generator, adding white noise to the RF signal.

5.3 Frame Error Estimation

Model 3, Section 3.5, involves an attempt to make decisions about optimal segmentation length based purely on knowledge obtained from the SS. The

two values, which are of interest, are $\Delta t_{rmit\,flag}$ and $\Delta t_{rmit\,timeout}$ from Equations 3.5.5 and 3.5.6. Data was collected from the TC-SDLP implementation's internal loop-back test.

At the SS, Δt, the delay between requesting segment insertion into the TS and actual insertion is logged. At the TS each failed frame was recorded. Each delay had to be compared to a baseline to obtain an estimation of whether it may be believed that a frame had been corrupted.

5.3.1 Single Errors Indicated by Retransmission Flag set in CLCW

Equation 3.5.5 gave $\Delta t_{rmit\,flag}$ as the lower bound of the single error interval. By Equation 3.5.3 $\Delta t_{rmit\,flag}$ is related to $\Delta t_{norm\,MAX}$ by an inequality. The first step in choosing a suitable value for $\Delta t_{rmit\,flag}$, is determining $\Delta t_{norm\,MAX}$. $\Delta t_{norm\,MAX}$ is used for for single error detection as this is the largest number of time a single transmission requires to be ACKed.

The Δt was transformed to a binary condition by comparing Δt to various test values, Δt_{test}. The number of binary occurrences were summed to 'Ê'. Ê being the number of estimated error occurrences.

The number of errors found in the data were transformed to the binary condition: "Did one or more errors occur". The number of binary error occurrences were summed to 'E'.

The ratio $\frac{\hat{E}}{E}$ indicates how many errors where estimated to have occured per real error, hence the relationship in Equation 5.3.1 may be reasoned. Equation 5.3.1 states: As the relationship between estimated errors per real errors tends to 1 all single retransmissions have been ACKed. Furthermore at the point where $\frac{\hat{E}}{E}$ reaches 1 the last error free frame has been ACKed. As the number of readings increase so does the accuracy of $\Delta t_{norm\,MAX}$.

$$\lim_{\Delta t_{test}^{-} = \rightarrow \Delta t_{norm\,MAX}} \frac{\hat{E}}{E} = 1 \tag{5.3.1}$$

Developing Equation 5.3.1 in terms of the mean of the $\frac{\hat{E}}{E}$ readings, $\frac{\mathrm{E}(\hat{E})}{\mathrm{E}(E)}$, leads to the same result. Investigating the data it was noted that, on startup, the error occurrences in the first frames resulted in retransmission times lower than $\Delta t_{norm\,MAX}$. This lead to a mean value of slightly less than 1. Figure 5.3.1 shows the mean and standard deviation of $\frac{\hat{E}}{E}$ at a frame length of 1022.

From Figure 5.3.1 it may be observed that the region, $3520, 5700$, has a constant value. The gradient is zero, because this is the region between the last frame, that took $\Delta t_{norm\,MAX}$ to be removed, and the first retransmitted frame to be removed from the sliding window data structure. Hence $\Delta t_{norm\,MAX}$ can be obtained by Equation 5.3.2. Similarly the value of $\Delta t_{MAX\,rmit\,flag}$ can be obtained by Equation 5.3.3

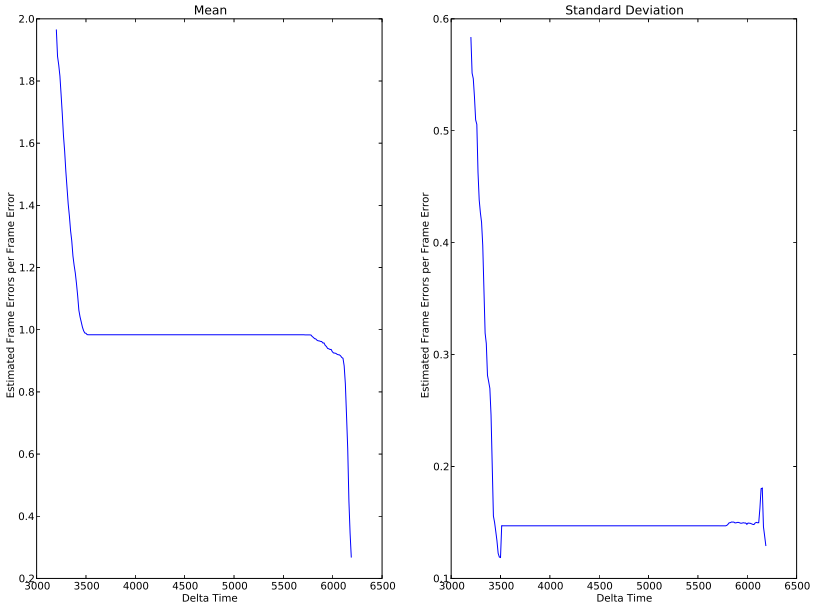


Figure 5.3.1: Mean and Standard Deviation of $\frac{\hat{E}}{E}$ with a Frame Length of 1022

$$\lim_{\Delta t_{test}^{-} = \rightarrow \Delta t_{norm\,MAX}} \frac{d}{d\frac{\hat{E}}{E}} \frac{\hat{E}}{E} = 0 \tag{5.3.2}$$

$$\lim_{\Delta t_{test}^{+} = \rightarrow \Delta t_{MAX\,rmit\,flag}} \frac{d}{d\frac{\hat{E}}{E}} \frac{\hat{E}}{E} = 0 \tag{5.3.3}$$

From Equation 5.3.2 and 5.3.3, the respective lower and upper bound of the interval, within which $\Delta t_{rmit\,flag}$ should fall, are located. These bounds of the interval, as well as the chosen value for the $\Delta t_{rmit\,flag}$, have, therefore, been located. Data was collected from four key indicator frame lengths, 266, 532, 760 and 1022. Data was obtained under various error conditions to ensure reasonable confidence in the resulting values. For completeness a control run from all remaining frame sizes was obtained. The results of the indicator frame lengths are summarized Table 5.3.2.

The $\Delta t_{rmit\,flag}$ was chosen to be in the middle of the bound. The averages of the points were obtained and are listed in Table 5.3.2. The line gradient and y-intercept, that minimizes the distance between the averages, was obtained by Equation 3.4.5 and 3.4.6. The line, expressed by Equation 5.3.4, is the decision threshold for a single retransmission.

Table 5.3.1: $\Delta t_{rmit\ flag}$ Interval

Frame Size	$\Delta t_{norm\ MAX}$	Average	$\Delta t_{MAX\ rmit\ flag}$
266	1070	1233.5	1397
532	1900	2405	2910
760	2940	3610	4280
1022	3520	4610	5700

$$\Delta t_{rmit\ flag} = 4.356 X_{seg\ len} + 39.015 \tag{5.3.4}$$

The indicator data, as well as control values, are shown in Figure 5.3.2. Figure 5.3.2 also shows $\Delta t_{rmit\ flag}$ calculated by Equation 5.3.4.

The decision threshold for a single error based on time delay has been formulated. A decision over a single error estimate can be made by comparing the observed time delay, Δt, to the expected time delay for a single error, $\Delta t_{rmit\ flag}$, which is calculated form the current segment length. This simple comparison may be achieved within an single if statement. Hence the operational complexity is very low.

5.3.2 Double Error Occurrences

Equation 3.5.4 gives the threshold $\Delta t_{rmit\ timeout}$ as an indication of multiple retransmissions. A similar approach to the one presented in Section 5.3.1 was conducted to obtain $\Delta t_{rmit\ timeout}$. The key difference lay in comparing Δt a double error occurrence.

By replacing the term $\Delta t_{norm\ MAX}$ with $\Delta t_{single\ rmit\ MAX}$ and $\Delta t_{MAX\ rmit\ flag}$ with $\Delta t_{MAX\ 2rmit\ flag}$ in Equations 5.3.2 and 5.3.3 respectively, these equations can be used to locate the double error threshold. Using the same four key values, as in Section 5.3.1, in conjunction with control data, the $\Delta t_{dble\ rmt}$ the $\Delta t_{2rmit\ flag}$ threshold was determined. With the method of least squares, a line was fitted to the data. This line equation for the $\Delta t_{2rmit\ flag}$ threshold is given as Equation 5.3.5 .

Table 5.3.2: $\Delta t_{rmit\ 2rmit\ flag}$ Interval

Frame Size	$\Delta t_{single\ rmit\ MAX}$	Average	$\Delta t_{MAX\ 2rmit\ flag}$
266	1890	2047.5	2205
532	3530	4085	4640
760	4900	4655	6610
1022	6550	7715	8880

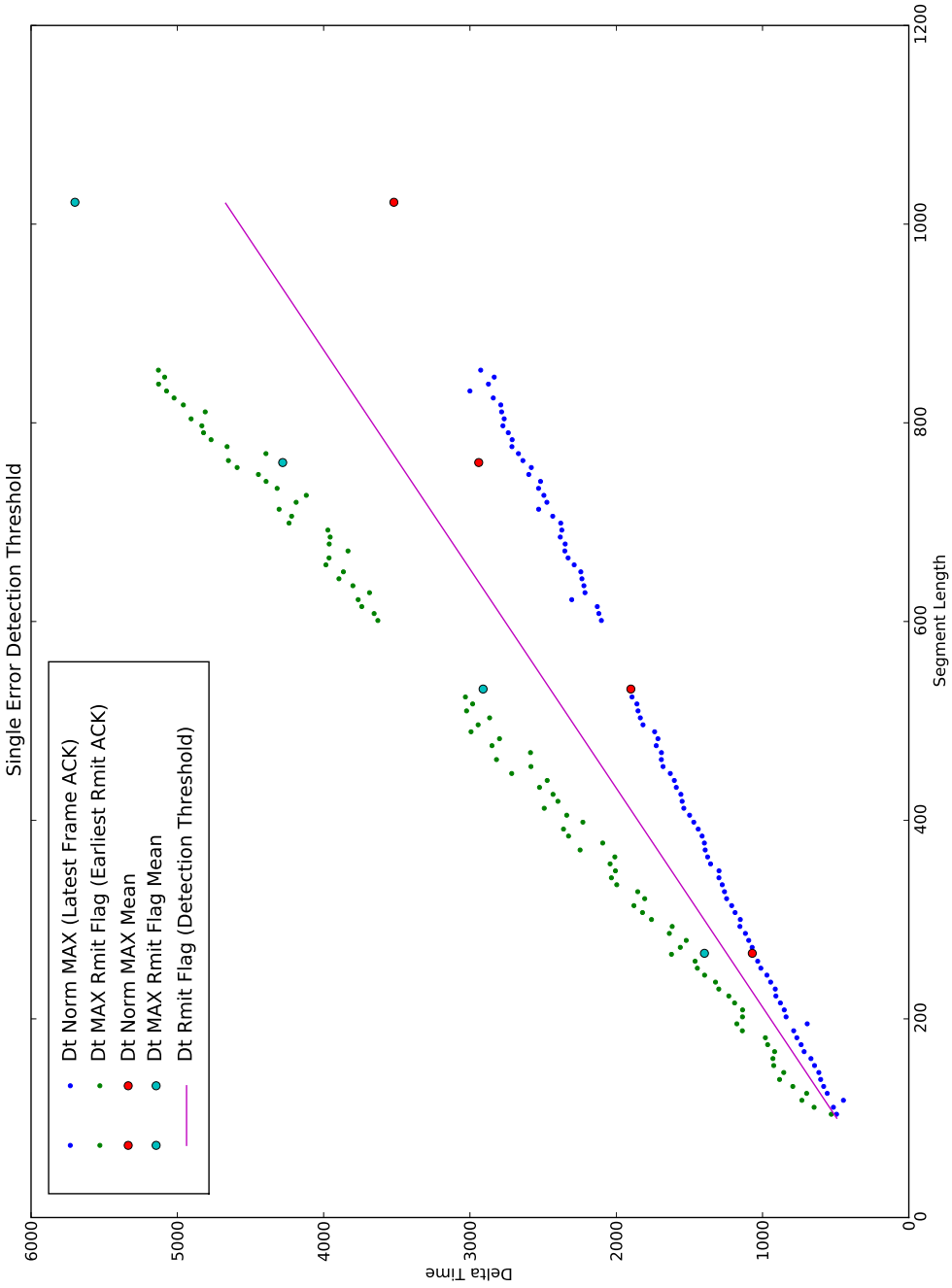


Figure 5.3.2: $\Delta t_{norm\,MAX}$,$\Delta t_{rmit\,flag}$ and $\Delta t_{MAX\,rmit\,flag}$ for Various Frame Lengths

$$\Delta t_{2rmit\,flag} = 5.668 X_{seg\,len} + 678.537 \tag{5.3.5}$$

5.4 System Results

The TC-SDLP implementation, discussed in Chapter 4, was used to test Model 3. Statistics on effective data throughput, the ratio of new data bits transmitted over total bits transmitted, were gathered. The number of new data bits is logged in the SS, while the number of total transmitted bits is logged in the SCCS.

Due to time constraints the amount of samples obtained from the system are limited. Notwithstanding aforementioned, enough data was collected to investigate trends, as discussed below.

5.4.1 Initial Results

The number of system errors was obtained by comparing the Δt between frame insertion and comparing the time taken to the frame error decision thresholds presented in Equation 5.3.4 and 5.3.5. The number of errors deduced from this comparison of Δt to the thresholds were used to update the frame error history queue.

The frame error history queue consists of 12 bytes. These 12 bytes are used to contain the 96 previous error occurrences which is the optimal frame history length, as per the discussion in Section 3.5. The results obtained were compared to the benchmark and the difference statistics are given in Table 5.4.1.

Table 5.4.1: Difference Between System Results and Simulation Results

BER	Number of Samples	Mean Difference	Standard Deviation of Difference
0.000005	46	-0.0729	0.0935
0.00005	35	-0.0861	0.0925
0.0005	20	-0.1355	0.0172
0.001	59	-0.1049	0.0051
0.003	29	-0.0243	0.0038
0.005	15	-0.0003	0.0039

The trend to be observed from the data at low BER, is that of moderately high deviation from the simulation benchmark results, and high standard deviation within system effective throughput. The trend at moderate BER is that of high deviation from the benchmark, and moderate standard devia-

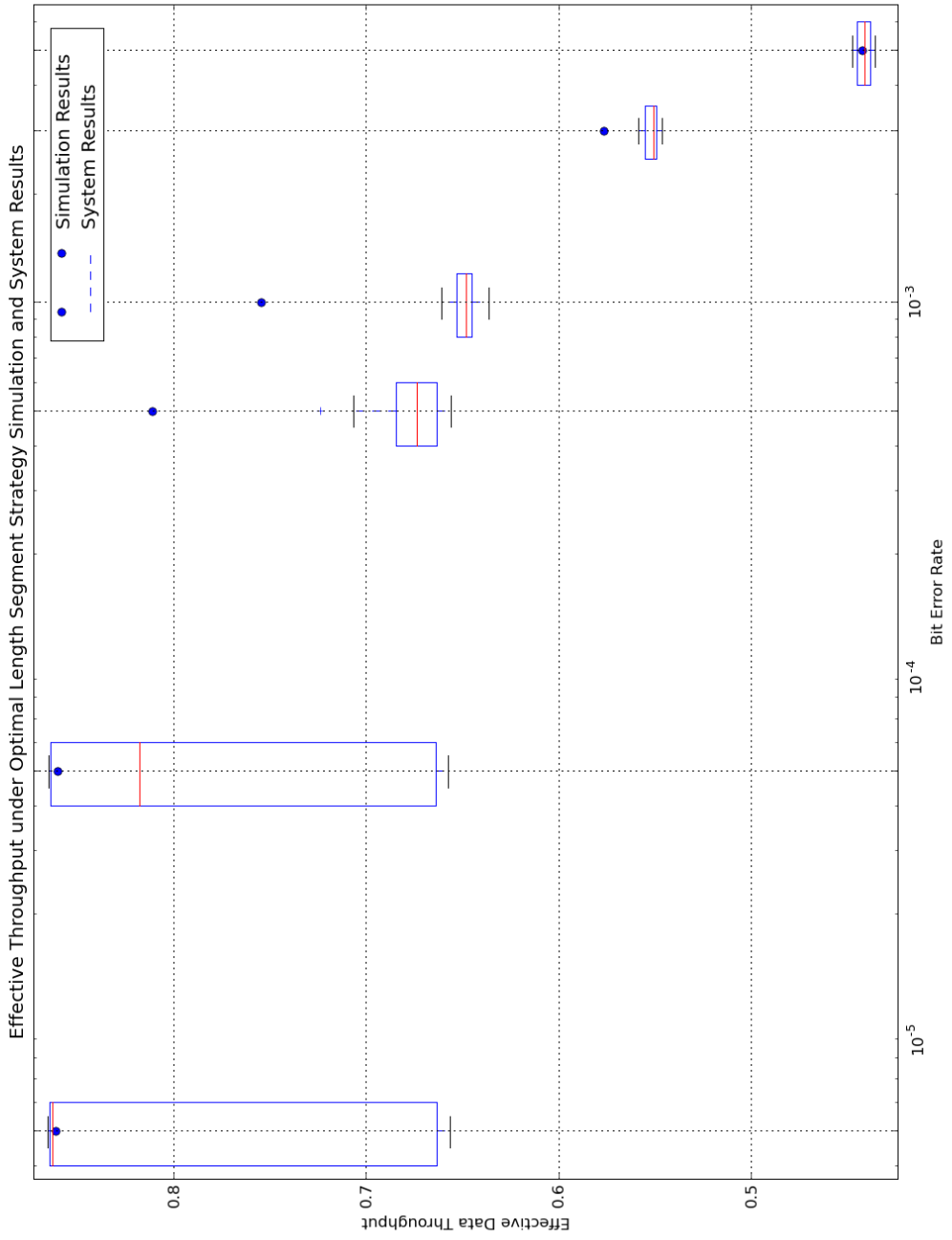


Figure 5.4.1: Box and Whiskers of System and Simulation Effective Throughput

tion from system effective throughput. At low BER the system results tend to follow the simulation results and they exhibit little internal deviation. Figure 5.4.2 presents this data in a box and whiskers graph. It should be noted, that the outliers are excluded from the box and whiskers graph and therefore the mean appears higher than in the data presented in Tabel 5.4.1.

The biggest discrepancy between the single error results lie in the regions of low BER. This region corresponds to the steep drop in the optimal segment length due to a slight change in FER See Figure 3.4.1.

5.4.2 Deviation Hypothesis

The deviation, of Table 5.4.1 between simulation and system results, may be caused by a "run-away effect". A small change in the FER causes a significant adjustment in the optimal segment length. If a double or triple error occured in a few consecutive frames, where the segments are long, then the next frame will be shortened considerably. Δt is the time taken to transmit the frame at the front of the sliding window queue. Assume that this was a long segment, but due to repeated frame losses, the frame to be inserted is short. A long Δt may then be associated with the short frame. Regardless of whether the frame failed or not the transmission time may be greater than the threshold obtained from the current frame, causing the system to assume an error occurred. This suspected retransmission may lead an even shorter next segment, causing another false belief that a retransmission occured. This process may continue until a frame length is reached where a large increase in FER is required to affect a change in optimal segment length. Once this lowest point is reached the frames will be shorter than the optimal segment length, meaning that there will be few errors leading in a rise in optimal segment length. Finally equilibrium might be reached again, but the decrease in frame length will have resulted in a decrease in effective data throughput.

Alternatively, the discrepancy might be caused by the differences between the simulation model and the real system. The key simplification might be that of ACKs not failing, although this simplification is expected to feature more in higher BER regions.

5.4.3 Introduction of Type 2 Error: Using Single Error Only

To test the hypothesis presented in Section 5.4.2, a type 2 error was introduced into the frame error detection process, a false negative. This type 2 error deliberately excluded all multiple error occurrences in a single frame, hence the F$\hat{\text{E}}$R is lowered. This trade-off of F$\hat{\text{E}}$R is hoped to circumnavigate the run-away effect described in the previous section. This final set of system results is presented in Table 5.4.2, where once again the data is compared to the benchmark.

Table 5.4.2: Difference Between System Results, with Type 2 Error, and Simulation Results

BER	Number of Samples	Mean Difference	Standard Deviation of Differnce
0.000005	31	0.0048	0.0005
0.00005	41	0.0000	0.0185
0.0005	24	-0.0738	0.0294
0.001	22	-0.0480	0.0123
0.003	27	-0.0030	0.0154
0.005	40	0.0086	0.0037

The results given in Table 5.4.2 are closer to the simulation results, insinuating that the run-away effect proposed in Section 5.4.2 did occur. A further point worth noting, is that Model 3, with a type 2 error method introduction, can successfully be used to determine a near optimal segmentation length strategy. It may be noted that as a minor positive side-effect Model 3 has been simplified by the introduction of the type 2 error.

5.5 Conclusion

This chapter investigated the results obtained from the simulation model, based on Model 2 of the optimal segment length strategy. The simulation results of effective throughput were compared to the expected results obtained for the mathematical Model 1. The similarity between these two sets of results presented much confidence in the validity and correctness of Model 2.

Since Model 2 was proven, attention was diverted to Model 3. Model 3 required a method to estimate error occurrences based on the time delay Δt. The two decision thresholds, one for single errors and one for multiple errors, were obtained using system optimal segment length strategy.

As the initial system results were not as good as might have been anticipated, an explanation to the results was presented. Owing to the introduction of a type 2 error, excluding double errors from Model 3, system results showed that near optimal segmentation length was indeed obtained.

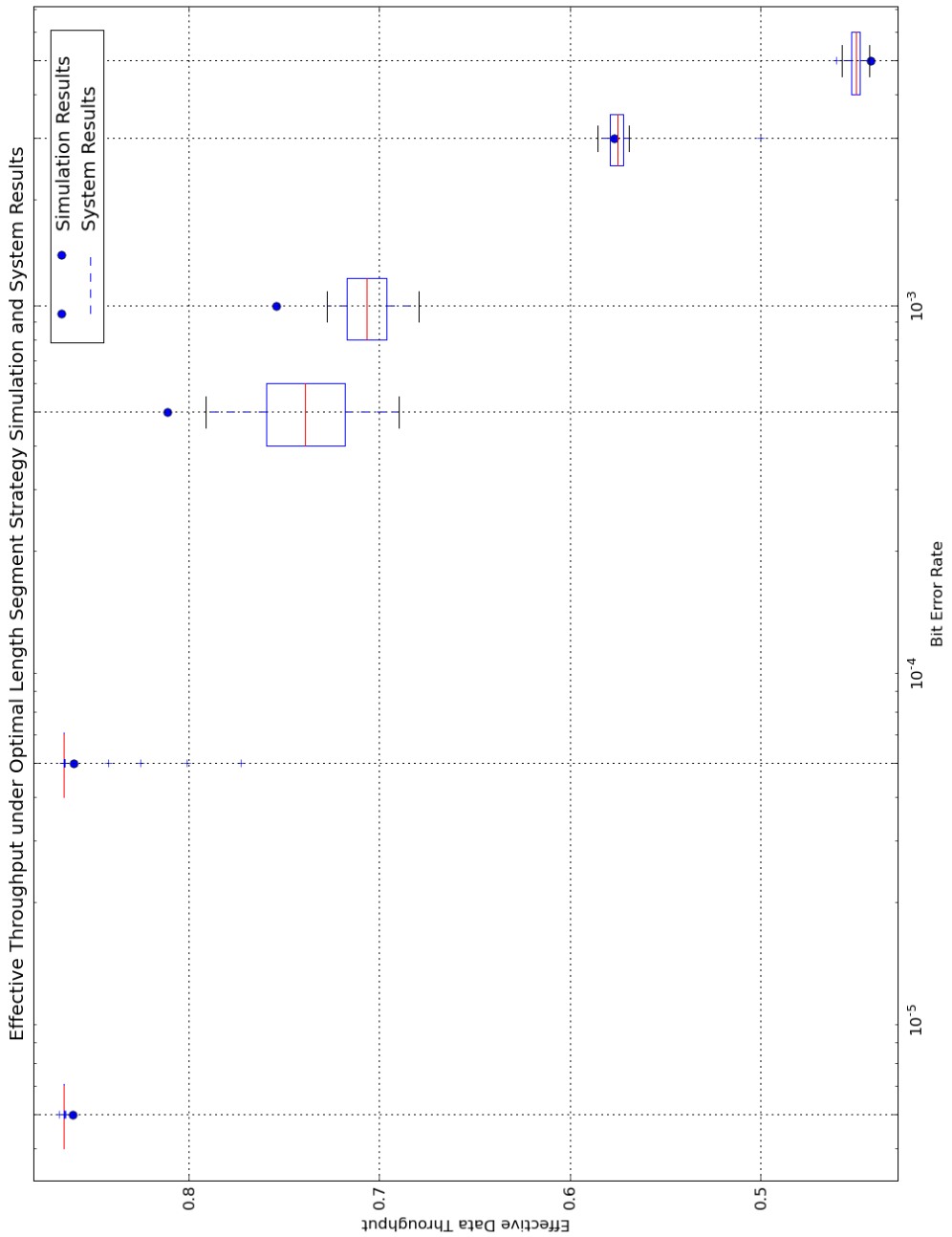


Figure 5.4.2: Box and Whiskers of System, with Type 2 Error, and Simulation Effective Throughput

Chapter 6

Conclusion, Recommendations and Future Work

This chapter draws conclusions based on the work done during this project. Furthermore, recommendations for future work emanating from work done in this project, are presented.

6.1 Summary of Work Done

This project comprised a large part of the telecommunications protocol for the MCSP LEO satellite. The scope of this project was the design and implementation of the datalink layer. The TC-SDLP, in conjunction with an FTP was chosen as to serve as the protocol stack. This shallow stack was chosen, because the satellite only serves as a bent pipe. Communication is required directly between satellite and ground station, i.e. no end to end routing is performed during satellite ground station communication. Any data file originating from outside the ground station is routed to the ground station via a suitable medium. Once the data file is successfully accepted at the ground station, it is stored there until it can be transmitted to the satellite. Conversely, a data file destined beyond the receiving ground station is first successfully transmitted to the ground station and then rerouted.

The link budget calculation showed that the communications link would profit from FEC. In this regard the cells of the TC-SDLP were encoded with a single error correcting BCH code.

During the course of the project an interesting scientific problem was identified, this being the maximising of the effective payload data throughput over the ground-space link by adaptively adjusting the segmentation strategy used in the TC-SS. This problem is based on the probability of any one cell failing in a TS frame, the start sequence being missed or the tail sequence being missed. These are all caused by the current BER and result in a FER. The payload data per total data in any TS frame times its FER, gave an

expression for the effective data throughput. The segment lengths that maximise the effective data throughput at any given BER could be found exactly. This method was dubbed Model 1.

Since the BER is not known by the TC-SDLP, the optimal length was presented in terms of FER. This method was called Model 2. Since the true FER cannot be known from a limited number of frame history samples, a F$\hat{E}$R was obtained. The best method by which this estimate could be obtained from the frame error history was discussed and a simple F$\hat{E}$R, based on lost frames per window, was found to give the best results.

As Model 2 makes use of knowledge contained within the TC-TS, but not in the TC-SS, a revised model, Model 3, was created. Model 3 uses information exclusively stemming from the TC-SS. Errors were estimated from time delays, which are accessible from within the TC-SS.

Model 1 was validated mathematically. A simulation model was built, which was based on Model 2. The results from the simulation compared very well to those obtained from Model 1. Thus Model 2 was shown to be a suitable method of obtaining the optimal segmentation length.

The TC-SDLP implementation was an almost complete implementation of the TC-SDLP standard. Model 3 was applied to the implementation and statistics were gained. These statistics revealed that under certain BER a runaway effect sometimes occured. This effect could be circumnavigated by artificially decreasing the error estimations. Error estimations were decreased by ignoring all retransmission errors. The results obtained from this modified version of Model 3, demonstrated that Model 3 can be used to achieve high effective data throughput.

Since Model 2 outperformed Model 3, it may be concluded that when possible, the frame errors should be obtained from the TC-TS. In cases where the TC-SS does not have access to this information contained in the TC-TS, a near optimal segmentation length strategy could be adopted.

6.2 Recommendations

The TC-SDLP implementation is still in prototype phase. The entire protocol was implemented to run on the SH-4 CPU. The bit wise operations, required to realise the FEC in the TC-SCCS, could be much more effectively performed in hardware.

Both FOP-1 and FARM-1 of TC-TS are state machines. State machines are very suitable for FPGAs, hence the TC-TS could be implemented in hardware.

Moving both the TC-SCCS and TC-TS to hardware would remove much of the processing strain of the current version of the TC-SDLP implementation on the SH-4.

6.3 Contributions

The core contribution of this thesis are:

- Introducing an algorithm to calculate frame length, which maximises effective data throughput, based on a satellite-ground link's BER.
- Refining the algorithm to use the FER metric present in the TC-SDLP.
- Obtaining a metric from content in the TC-SS, whereby frame losses can be estimated .
- Identifying the best method to obtain the F$\hat{E}$R from a history of frame losses.
- Building a simulation model to test and verify the algorithm.
- Implementing the TC-SDLP standard in software.
- Using frame loss estimation as a basis to obtain the F$\hat{E}$R, with the F$\hat{E}$R providing the optimal segment length for the TC-SDLP implementation.

6.4 Future Work

The work presented in this project is different to most contemporary wireless research, in that the wireless link was optimised for a single hop, i.e. device to device. Single hop links, such as blue-tooth, may benefit greatly from the adaptive frame length technology presented in this work.

The blue-tooth standard features repetition as a method of FEC. Using adaptive frame length with a simple BCH code, as was done in this project, may improve effective data throughput over a such a link. The principles obtained in this work could be applied to a device to device link standard, leading to a device to device protocol more efficient than the current blue-tooth or similar standards.

Bibliography

[1] European Centre for Space Standardisation, "ECSS-S-ST-00C - ECSS system, Description, implementation and general requirements," ECSS, Tech. Rep., July 2008.

[2] European Space Agency, 2008, http://www.esa.int/SPECIALS/ESOC/SEMU2CW4QWD_0.html.

[3] National Aeronautics and Space Administration, "Orbital Debris Quarterly News," April 2009, Volume 13, Issue 2.

[4] European Space Agency, 2008, http://www.esa.int/.

[5] National Aeronautics and Space Administration, 2008, http://www.nasa.gov/.

[6] European Centre for Space Standardisation, "ECSS-P-001B," ECSS, Tech. Rep., July 2004.

[7] ——, "ECSS-E-10-Part6A," ECSS, Tech. Rep., October 2005.

[8] ——, "ECSS-M-10C," ECSS, Tech. Rep., July 2008.

[9] ——, "ECSS-E-ST-40C," ECSS, Tech. Rep., March 2009.

[10] Consultative Committee for Space Data Systems, 2009, http://public.ccsds.org/default.aspx.

[11] R. E. Ziemer and W. H. Tranter, *Principles of Communications*, 5th ed. John Wiley & Sons, inc, 2002.

[12] S. Lin and J. Daniel J. Costello, *Error Control Coding: Fundamentals and Applications*, 1st ed. Prentice-Hall, Inc, 1983.

[13] P. Sweeney, *Error Control Coding from Theory to Practice*, 1st ed. John Wiley & Sons, 2002.

[14] W. W. Peterson and J. E. J. Weldon, *Error-Correcting Codes*, 2nd ed. The MIT Press, 1972.

[15] V. G. Cerf and R. E. Kahn, "A protocol for packet network intercommunication," *IEEE Transactions on Communications*, May 1974.

[16] H. Zimmermann, "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications*, April 1980.

[17] European Centre for Space Standardisation, "ECSS-E-HB-50A," November 2008.

[18] A. Cooke, "Rural e-mail system for the sumbandila satellite," Master's thesis, Stellenbosch University, 2007.

[19] S. Holden, *Python Web Programing*, 1st ed. New Riders, 2002.

[20] Network Working Group, *File Transfer Protocol (FTP)*, October 1985, RFC 959.

[21] Managment Council of the CCSDS, "CCSDS 130.0-G-2," CCSDS, Tech. Rep., December 2007.

[22] European Centre for Space Standardisation, "ECSS-E-50-03A," ECSS, Tech. Rep., November 2007.

[23] ——, "ECSS-E-50-01A," ECSS, Tech. Rep., November 2007.

[24] Managment Council of the CCSDS, "CCSDS 732.0-B-2," CCSDS, Tech. Rep., July 2006.

[25] European Centre for Space Standardisation, "ECSS-E-50-04A," ECSS, Tech. Rep., November 2007.

[26] A. Elahi and M. Elahi, *Data, Network & Internet Communications Technology*, 1st ed. Thomson Delmar Learning, 2006.

[27] CISCO, 2008, `"http://www.cisco.com/univercd/cc/td/doc/product/atm/c8540/12_1/pereg_1/atm_tech/basics.htm"`.

[28] Managment Council of the CCSDS, "CCSDS 210.0-G-1," CCSDS, Tech. Rep., August 2007.

[29] ——, "CCSDS 211.0-B-4," CCSDS, Tech. Rep., July 2006.

[30] ——, "CCSDS 211.2-B-1," CCSDS, Tech. Rep., April 2003.

[31] ——, "CCSDS 211.1-B-3," CCSDS, Tech. Rep., March 2006.

[32] IEEE 802.3 ETHERNET WORKING GROUP. [Online]. Available: http://www.ieee802.org/3/

[33] F. Calí, M. Conti, and E. Gregori, "Ieee 802.11 wireless lan: Capacity analysis and protocol enhancement," *IEEE Transactions on Communications*, April 1998.

[34] A. Vasan and A. U. Shankar, "An empirical characterization of instantaneous throughput in 802.11b wlans," *Technical Reports from UMIACS*, October 2002.

[35] R. P. F. Hoefel, "A mac and phy cross-layer analytical model for the goodput and delay of ieee 802.11a networks operating under basic access and rts/cts dcf schemes," *Journal of Communications*, September 2006.

[36] N. Celandroni and F. Potortí, "Maximising single connection tcp goodput by trading bandwidth for ber," *International Journal of Communication Systems*, February 2003.

[37] G. van Rossum and J. Fred L. Drake, *Python Reference Manual*, 2nd ed., December 2003.

[38] ——, *Python Library Reference*, 2nd ed., April 2002.

[39] Python Software Foundation, 2008, `http://www.python.org`.

[40] D. Collett, *Modeling Binary Data*, 2nd ed. Chapman & Hall/CRC, 2003.

[41] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," *Department of Computer Science University of North Carolina at Chapel Hill*, July 2006.

[42] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 3rd ed. McGraw-Hill, 2000.

[43] M. Galassi, J. Davies, J. Theiler, B. Gough, J. Jungman, M. Booth, and F. Rossi, *GNU Scientific Library*, 1st ed., February 2008.

[44] S. Furr, "What is real time and why do i need it?" QNX Software Systems Ltd., Tech. Rep., 2002.

Appendix A

Preliminary SNR Table

Table A.1: Preliminary SNR calculation, based on an Omni-directional GS antenna

	0^o	5^o	10^o	15^o	20^o	25^o	30^o	35^o	40^o	45^o
0^o										
5^o										
10^o										
15^o										-18.914
20^o								-20.463	-17.839	-16.144
25^o							-19.825	-17.267	-15.509	-14.112
30^o						-19.825	-17.085	-15.203	-13.695	-12.429
35^o					-20.463	-17.267	-15.203	-13.558	-12.170	-10.972
40^o					-17.839	-15.509	-13.695	-12.170	-10.848	-9.687
45^o				-18.914	-16.144	-14.112	-12.429	-10.972	-9.687	-8.543
50^o			-21.059	-17.175	-14.831	-12.953	-11.346	-9.930	-8.663	-7.524
55^o			-18.797	-15.902	-13.764	-11.978	-10.418	-9.024	-7.764	-6.621
60^o			-17.453	-14.904	-12.887	-11.159	-9.627	-8.244	-6.983	-5.829
65^o		-19.974	-16.476	-14.112	-12.170	-10.479	-8.963	-7.584	-6.318	-5.150
70^o		-18.797	-15.743	-13.490	-11.597	-9.930	-8.423	-7.043	-5.768	-4.584
75^o		-18.040	-15.203	-13.020	-11.159	-9.506	-8.004	-6.621	-5.336	-4.137
80^o		-17.547	-14.831	-12.689	-10.848	-9.204	-7.704	-6.318	-5.025	-3.813
85^o		-17.267	-14.612	-12.494	-10.663	-9.024	-7.524	-6.135	-4.837	-3.616
90^o	-22.798	-17.175	-14.539	-12.429	-10.602	-8.963	-7.464	-6.074	-4.774	-3.550

	50^o	55^o	60^o	65^o	70^o	75^o	80^o	85^o	90^o
0^o									-22.798
5^o				-19.974	-18.797	-18.040	-17.547	-17.267	-17.175
10^o	-21.059	-18.797	-17.453	-16.476	-15.743	-15.203	-14.831	-14.612	-14.539
15^o	-17.175	-15.902	-14.904	-14.112	-13.490	-13.020	-12.689	-12.494	-12.429
20^o	-14.831	-13.764	-12.887	-12.170	-11.597	-11.159	-10.848	-10.663	-10.602
25^o	-12.953	-11.978	-11.159	-10.479	-9.930	-9.506	-9.204	-9.024	-8.963
30^o	-11.346	-10.418	-9.627	-8.963	-8.423	-8.004	-7.704	-7.524	-7.464
35^o	-9.930	-9.024	-8.244	-7.584	-7.043	-6.621	-6.318	-6.135	-6.074
40^o	-8.663	-7.764	-6.983	-6.318	-5.768	-5.336	-5.025	-4.837	-4.774
45^o	-7.524	-6.621	-5.829	-5.150	-4.584	-4.137	-3.813	-3.616	-3.550
50^o	-6.500	-5.584	-4.774	-4.073	-3.484	-3.015	-2.672	-2.463	-2.393
55^o	-5.584	-4.648	-3.813	-3.083	-2.463	-1.965	-1.598	-1.372	-1.296
60^o	-4.774	-3.813	-2.947	-2.181	-1.523	-0.986	-0.586	-0.337	-0.253
65^o	-4.073	-3.083	-2.181	-1.372	-0.667	-0.081	0.364	0.498	0.519
70^o	-3.484	-2.463	-1.523	-0.667	0.094	0.519	0.626	0.689	0.710
75^o	-3.015	-1.965	-0.986	-0.081	0.519	0.668	0.773	0.835	0.856
80^o	-2.672	-1.598	-0.586	0.364	0.626	0.773	0.876	0.938	0.959
85^o	-2.463	-1.372	-0.337	0.498	0.689	0.835	0.938	1.000	1.020
90^o	-2.393	-1.296	-0.253	0.519	0.710	0.856	0.959	1.020	1.041

Appendix B

BCH Syndromes

Table B.1: Syndrome S1 and S2 resulting from a single error polynomial $e(X) = X^i$ where $0 \leq i < n$

Power	Polynamial	Binary	Hexadecimal
$e(\alpha^0)$	1	0b00000001	0x1
$e((\alpha^0)^2)$	1	0b00000001	0x1
$e(\alpha^1)$	$+\alpha$	0b00000010	0x2
$e((\alpha^1)^2)$	$+\alpha^2$	0b00000100	0x4
$e(\alpha^2)$	$+\alpha^2$	0b00000100	0x4
$e((\alpha^2)^2)$	$+\alpha^4$	0b00010000	0x10
$e(\alpha^3)$	$+\alpha^3$	0b00001000	0x8
$e((\alpha^3)^2)$	$1+\alpha$	0b00000011	0x3
$e(\alpha^4)$	$+\alpha^4$	0b00010000	0x10
$e((\alpha^4)^2)$	$+\alpha^2+\alpha^3$	0b00001100	0xc
$e(\alpha^5)$	$+\alpha^5$	0b00100000	0x20
$e((\alpha^5)^2)$	$+\alpha^4+\alpha^5$	0b00110000	0x30
$e(\alpha^6)$	$1+\alpha$	0b00000011	0x3
$e((\alpha^6)^2)$	$1+\alpha^2$	0b00000101	0x5
$e(\alpha^7)$	$+\alpha+\alpha^2$	0b00000110	0x6
$e((\alpha^7)^2)$	$+\alpha^2+\alpha^4$	0b00010100	0x14
$e(\alpha^8)$	$+\alpha^2+\alpha^3$	0b00001100	0xc
$e((\alpha^8)^2)$	$1+\alpha+\alpha^4$	0b00010011	0x13
$e(\alpha^9)$	$+\alpha^3+\alpha^4$	0b00011000	0x18
$e((\alpha^9)^2)$	$1+\alpha+\alpha^2+\alpha^3$	0b00001111	0xf
$e(\alpha^{10})$	$+\alpha^4+\alpha^5$	0b00110000	0x30
$e((\alpha^{10})^2)$	$+\alpha^2+\alpha^3+\alpha^4+\alpha^5$	0b00111100	0x3c
$e(\alpha^{11})$	$1+\alpha+\alpha^5$	0b00100011	0x23
$e((\alpha^{11})^2)$	$1+\alpha^2+\alpha^4+\alpha^5$	0b00110101	0x35
$e(\alpha^{12})$	$1+\alpha^2$	0b00000101	0x5
$e((\alpha^{12})^2)$	$1+\alpha^4$	0b00010001	0x11
$e(\alpha^{13})$	$+\alpha+\alpha^3$	0b00001010	0xa
$e((\alpha^{13})^2)$	$1+\alpha+\alpha^2$	0b00000111	0x7
$e(\alpha^{14})$	$+\alpha^2+\alpha^4$	0b00010100	0x14
$e((\alpha^{14})^2)$	$+\alpha^2+\alpha^3+\alpha^4$	0b00011100	0x1c
$e(\alpha^{15})$	$+\alpha^3+\alpha^5$	0b00101000	0x28
$e((\alpha^{15})^2)$	$1+\alpha+\alpha^4+\alpha^5$	0b00110011	0x33

$e(\alpha^{16})$	$1+\alpha+\alpha^4$	0b00010011	0x13
$e((\alpha^{16})^2)$	$1+\alpha^3$	0b00001001	0x9
$e(\alpha^{17})$	$\alpha+\alpha^2+\alpha^5$	0b00100110	0x26
$e((\alpha^{17})^2)$	$\alpha^2+\alpha^5$	0b00100100	0x24
$e(\alpha^{18})$	$1+\alpha+\alpha^2+\alpha^3$	0b00001111	0xf
$e((\alpha^{18})^2)$	$\alpha+\alpha^2+\alpha^4$	0b00010110	0x16
$e(\alpha^{19})$	$\alpha+\alpha^2+\alpha^3+\alpha^4$	0b00011110	0x1e
$e((\alpha^{19})^2)$	$1+\alpha+\alpha^3+\alpha^4$	0b00011011	0x1b
$e(\alpha^{20})$	$\alpha^2+\alpha^3+\alpha^4+\alpha^5$	0b00111100	0x3c
$e((\alpha^{20})^2)$	$1+\alpha+\alpha^2+\alpha^3+\alpha^5$	0b00101111	0x2f
$e(\alpha^{21})$	$1+\alpha+\alpha^3+\alpha^4+\alpha^5$	0b00111011	0x3b
$e((\alpha^{21})^2)$	$\alpha+\alpha^3+\alpha^4+\alpha^5$	0b00111010	0x3a
$e(\alpha^{22})$	$1+\alpha^2+\alpha^4+\alpha^5$	0b00110101	0x35
$e((\alpha^{22})^2)$	$1+\alpha^2+\alpha^3+\alpha^5$	0b00101101	0x2d
$e(\alpha^{23})$	$1+\alpha^3+\alpha^5$	0b00101001	0x29
$e((\alpha^{23})^2)$	$\alpha+\alpha^4+\alpha^5$	0b00110010	0x32
$e(\alpha^{24})$	$1+\alpha^4$	0b00010001	0x11
$e((\alpha^{24})^2)$	$1+\alpha^2+\alpha^3$	0b00001101	0xd
$e(\alpha^{25})$	$\alpha+\alpha^5$	0b00100010	0x22
$e((\alpha^{25})^2)$	$\alpha^2+\alpha^4+\alpha^5$	0b00110100	0x34
$e(\alpha^{26})$	$1+\alpha+\alpha^2$	0b00000111	0x7
$e((\alpha^{26})^2)$	$1+\alpha^2+\alpha^4$	0b00010101	0x15
$e(\alpha^{27})$	$\alpha+\alpha^2+\alpha^3$	0b00001110	0xe
$e((\alpha^{27})^2)$	$1+\alpha+\alpha^2+\alpha^4$	0b00010111	0x17
$e(\alpha^{28})$	$\alpha^2+\alpha^3+\alpha^4$	0b00011100	0x1c
$e((\alpha^{28})^2)$	$1+\alpha+\alpha^2+\alpha^3+\alpha^4$	0b00011111	0x1f
$e(\alpha^{29})$	$\alpha^3+\alpha^4+\alpha^5$	0b00111000	0x38
$e((\alpha^{29})^2)$	$1+\alpha+\alpha^2+\alpha^3+\alpha^4+\alpha^5$	0b00111111	0x3f
$e(\alpha^{30})$	$1+\alpha+\alpha^4+\alpha^5$	0b00110011	0x33
$e((\alpha^{30})^2)$	$1+\alpha^3+\alpha^4+\alpha^5$	0b00111001	0x39
$e(\alpha^{31})$	$1+\alpha^2+\alpha^5$	0b00100101	0x25
$e((\alpha^{31})^2)$	$1+\alpha^5$	0b00100001	0x21
$e(\alpha^{32})$	$1+\alpha^3$	0b00001001	0x9
$e((\alpha^{32})^2)$	α	0b00000010	0x2
$e(\alpha^{33})$	$\alpha+\alpha^4$	0b00010010	0x12
$e((\alpha^{33})^2)$	α^3	0b00001000	0x8

$e(\alpha^{34})$			$+\alpha^2$			$+\alpha^5$	0b00100100	0x24
$e((\alpha^{34})^2)$						$+\alpha^5$	0b00100000	0x20
$e(\alpha^{35})$	1	$+\alpha$		$+\alpha^3$			0b00001011	0xb
$e((\alpha^{35})^2)$		$+\alpha$	$+\alpha^2$				0b00000110	0x6
$e(\alpha^{36})$		$+\alpha$	$+\alpha^2$		$+\alpha^4$		0b00010110	0x16
$e((\alpha^{36})^2)$				$+\alpha^3$	$+\alpha^4$		0b00011000	0x18
$e(\alpha^{37})$			$+\alpha^2$	$+\alpha^3$		$+\alpha^5$	0b00101100	0x2c
$e((\alpha^{37})^2)$	1	$+\alpha$				$+\alpha^5$	0b00100011	0x23
$e(\alpha^{38})$	1	$+\alpha$		$+\alpha^3$	$+\alpha^4$		0b00011011	0x1b
$e((\alpha^{38})^2)$		$+\alpha$		$+\alpha^3$			0b00001010	0xa
$e(\alpha^{39})$		$+\alpha$	$+\alpha^2$		$+\alpha^4$	$+\alpha^5$	0b00110110	0x36
$e((\alpha^{39})^2)$				$+\alpha^3$		$+\alpha^5$	0b00101000	0x28
$e(\alpha^{40})$	1	$+\alpha$	$+\alpha^2$	$+\alpha^3$		$+\alpha^5$	0b00101111	0x2f
$e((\alpha^{40})^2)$		$+\alpha$	$+\alpha^2$			$+\alpha^5$	0b00100110	0x26
$e(\alpha^{41})$	1		$+\alpha^2$	$+\alpha^3$	$+\alpha^4$		0b00011101	0x1d
$e((\alpha^{41})^2)$		$+\alpha$	$+\alpha^2$	$+\alpha^3$	$+\alpha^4$		0b00011110	0x1e
$e(\alpha^{42})$		$+\alpha$		$+\alpha^3$	$+\alpha^4$	$+\alpha^5$	0b00111010	0x3a
$e((\alpha^{42})^2)$	1	$+\alpha$		$+\alpha^3$	$+\alpha^4$	$+\alpha^5$	0b00111011	0x3b
$e(\alpha^{43})$	1	$+\alpha$	$+\alpha^2$		$+\alpha^4$	$+\alpha^5$	0b00110111	0x37
$e((\alpha^{43})^2)$	1			$+\alpha^3$		$+\alpha^5$	0b00101001	0x29
$e(\alpha^{44})$	1		$+\alpha^2$	$+\alpha^3$		$+\alpha^5$	0b00101101	0x2d
$e((\alpha^{44})^2)$		$+\alpha$				$+\alpha^5$	0b00100010	0x22
$e(\alpha^{45})$	1			$+\alpha^3$	$+\alpha^4$		0b00011001	0x19
$e((\alpha^{45})^2)$		$+\alpha$	$+\alpha^2$	$+\alpha^3$			0b00001110	0xe
$e(\alpha^{46})$		$+\alpha$			$+\alpha^4$	$+\alpha^5$	0b00110010	0x32
$e((\alpha^{46})^2)$				$+\alpha^3$	$+\alpha^4$	$+\alpha^5$	0b00111000	0x38
$e(\alpha^{47})$	1	$+\alpha$	$+\alpha^2$			$+\alpha^5$	0b00100111	0x27
$e((\alpha^{47})^2)$	1		$+\alpha^2$			$+\alpha^5$	0b00100101	0x25
$e(\alpha^{48})$	1		$+\alpha^2$	$+\alpha^3$			0b00001101	0xd
$e((\alpha^{48})^2)$		$+\alpha$			$+\alpha^4$		0b00010010	0x12
$e(\alpha^{49})$		$+\alpha$		$+\alpha^3$	$+\alpha^4$		0b00011010	0x1a
$e((\alpha^{49})^2)$	1	$+\alpha$		$+\alpha^3$			0b00001011	0xb
$e(\alpha^{50})$			$+\alpha^2$		$+\alpha^4$	$+\alpha^5$	0b00110100	0x34
$e((\alpha^{50})^2)$			$+\alpha^2$	$+\alpha^3$		$+\alpha^5$	0b00101100	0x2c
$e(\alpha^{51})$	1	$+\alpha$		$+\alpha^3$		$+\alpha^5$	0b00101011	0x2b
$e((\alpha^{51})^2)$		$+\alpha$	$+\alpha^2$		$+\alpha^4$	$+\alpha^5$	0b00110110	0x36

$e(\alpha^{52})$	1		$+\alpha^2$		$+\alpha^4$			0b00010101	0x15
$e((\alpha^{52})^2)$	1		$+\alpha^2$	$+\alpha^3$	$+\alpha^4$			0b00011101	0x1d
$e(\alpha^{53})$		$+\alpha$		$+\alpha^3$		$+\alpha^5$		0b00101010	0x2a
$e((\alpha^{53})^2)$	1	$+\alpha$	$+\alpha^2$		$+\alpha^4$	$+\alpha^5$		0b00110111	0x37
$e(\alpha^{54})$	1	$+\alpha$	$+\alpha^2$		$+\alpha^4$			0b00010111	0x17
$e((\alpha^{54})^2)$	1			$+\alpha^3$	$+\alpha^4$			0b00011001	0x19
$e(\alpha^{55})$		$+\alpha$	$+\alpha^2$	$+\alpha^3$		$+\alpha^5$		0b00101110	0x2e
$e((\alpha^{55})^2)$	1	$+\alpha$	$+\alpha^2$			$+\alpha^5$		0b00100111	0x27
$e(\alpha^{56})$	1	$+\alpha$	$+\alpha^2$	$+\alpha^3$	$+\alpha^4$			0b00011111	0x1f
$e((\alpha^{56})^2)$		$+\alpha$		$+\alpha^3$	$+\alpha^4$			0b00011010	0x1a
$e(\alpha^{57})$		$+\alpha$	$+\alpha^2$	$+\alpha^3$	$+\alpha^4$	$+\alpha^5$		0b00111110	0x3e
$e((\alpha^{57})^2)$	1	$+\alpha$		$+\alpha^3$		$+\alpha^5$		0b00101011	0x2b
$e(\alpha^{58})$	1	$+\alpha$	$+\alpha^2$	$+\alpha^3$	$+\alpha^4$	$+\alpha^5$		0b00111111	0x3f
$e((\alpha^{58})^2)$		$+\alpha$		$+\alpha^3$		$+\alpha^5$		0b00101010	0x2a
$e(\alpha^{59})$	1		$+\alpha^2$	$+\alpha^3$	$+\alpha^4$	$+\alpha^5$		0b00111101	0x3d
$e((\alpha^{59})^2)$		$+\alpha$	$+\alpha^2$	$+\alpha^3$		$+\alpha^5$		0b00101110	0x2e
$e(\alpha^{60})$	1			$+\alpha^3$	$+\alpha^4$	$+\alpha^5$		0b00111001	0x39
$e((\alpha^{60})^2)$		$+\alpha$	$+\alpha^2$	$+\alpha^3$	$+\alpha^4$	$+\alpha^5$		0b00111110	0x3e
$e(\alpha^{61})$	1				$+\alpha^4$	$+\alpha^5$		0b00110001	0x31
$e((\alpha^{61})^2)$	1		$+\alpha^2$	$+\alpha^3$	$+\alpha^4$	$+\alpha^5$		0b00111101	0x3d
$e(\alpha^{62})$	1					$+\alpha^5$		0b00100001	0x21
$e((\alpha^{62})^2)$	1				$+\alpha^4$	$+\alpha^5$		0b00110001	0x31

Appendix C

Simulation Results

Table C.1: Simulation Results at various BER

BER	0.000000	0.000010	0.000020	0.000030	0.000040	0.000050	0.000060	0.000070	0.000080	0.000090
Mean	0.860780	0.860754	0.860678	0.860550	0.858171	0.860138	0.859853	0.859514	0.859122	0.858676
Std. Dev.	0.000000	0.000005	0.000009	0.000014	0.000049	0.000023	0.000029	0.000034	0.000039	0.000044

BER	0.000100	0.000200	0.000300	0.000400	0.000500	0.000600	0.000700	0.000800	0.000900	0.001000
Mean	0.858171	0.850180	0.838091	0.824438	0.810917	0.798124	0.786175	0.774930	0.764267	0.753990
Std. Dev.	0.000049	0.000106	0.000142	0.000167	0.000183	0.000196	0.000215	0.000218	0.000225	0.000235

BER	0.002000	0.003000	0.004000	0.005000	0.006000	0.007000	0.008000	0.009000	0.010000	0.020000
Mean	0.659555	0.576189	0.504148	0.441649	0.387118	0.339410	0.297416	0.260836	0.229293	0.060316
Std. Dev.	0.000259	0.000290	0.000294	0.000301	0.000291	0.000281	0.000269	0.000253	0.000238	0.000133

BER	0.030000	0.040000	0.050000
Mean	0.017596	0.004449	0.000905
Std. Dev.	0.000060	0.000034	0.000040

Appendix D

"Core Obsolete" Warning

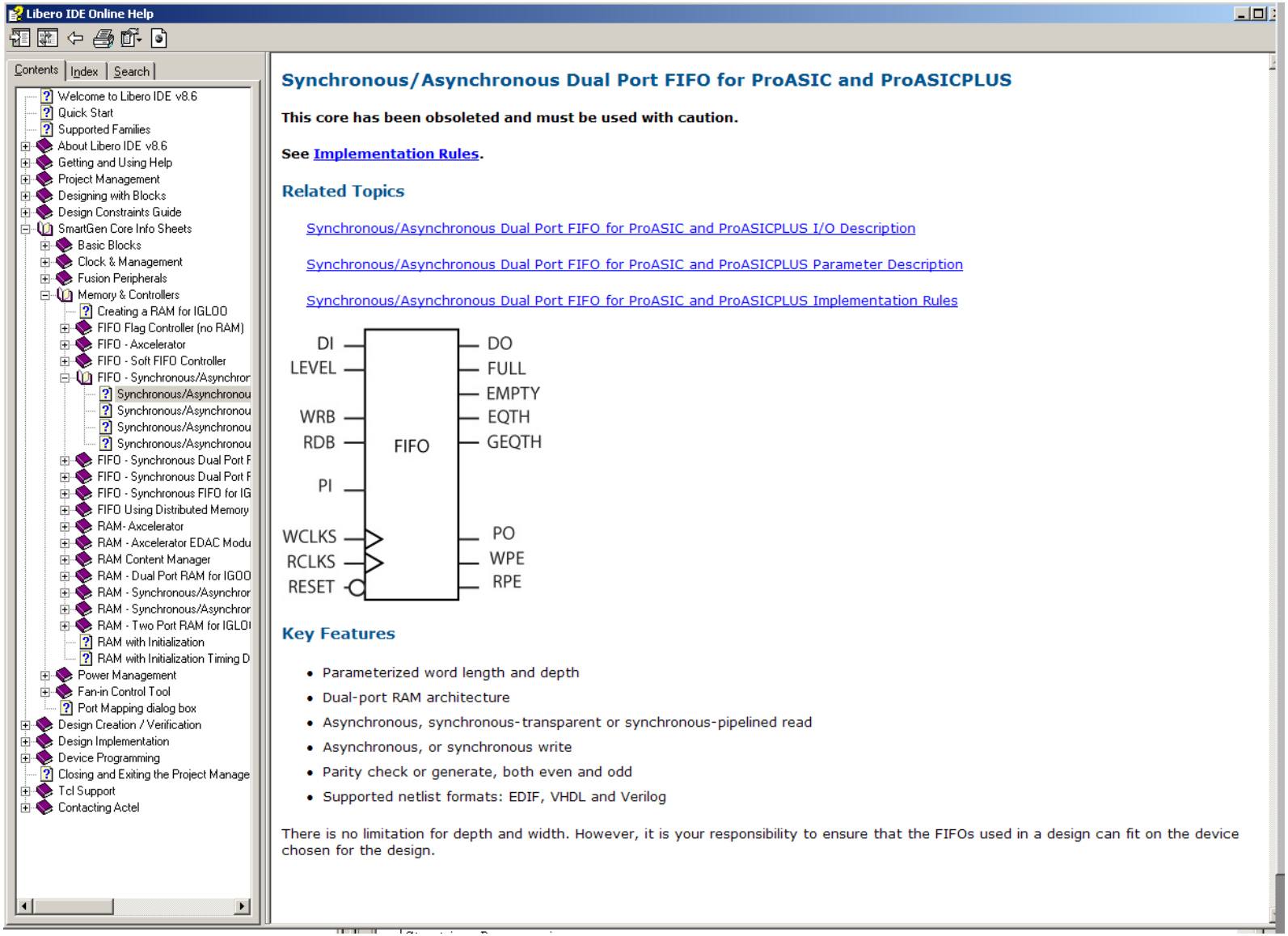


Figure D.0.1: Libero IDE Core Obsolete Warning

Appendix E

TC-SDLP Implementation Code